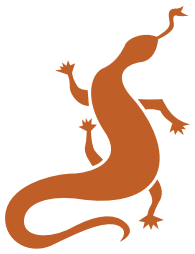


Goanna Studio User Guide

Version 3.6.4
Eclipse Edition

November 23, 2015

© 2008-2015, Red Lizard Software

Copyright © 2008-2015 Red Lizard Software
All rights reserved.

This document, as well as the software described in it, is provided under license and may only be used or copied in accordance with the terms of such license. The information contained herein is the property of NICTA and is made available under license to Red Lizard Software. It is confidential information as between the Recipient and Red Lizard Software and remains the exclusive property of NICTA. No part of this documentation may be copied, translated, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of NICTA.
NICTA does not warrant that this document is error-free.

Red Lizard Software
Australian Technology Park
Level 5, 13 Garden Street
Eveleigh NSW 2015
Australia

Web: http://www.redlizards.com
Support: support@redlizards.com

Contents

1 System Requirements **5**

1.1 Operating Systems . . . 5

1.1.1 Microsoft Windows . . . 5

1.1.2 Linux . . . 5

1.1.3 Other Requirements . . . 5

1.2 IDE Requirements . . . 6

1.3 Hardware Requirements . . . 6

1.4 Supported Compilers . . . 7

1.4.1 A Word On C99 and C++11 Support . . . 7

1.4.2 A Word On Compiler-Specific Syntax Extensions . . . 7

1.4.3 Analog Devices C/C++ Compiler for SHARC Processors (`cc21k` Dialect) . . . 7

1.4.4 ARM C/C++ Compiler (`armcc` Dialect) . . . 8

1.4.5 Cosmic Software C Cross Compiler (`cosmic` Dialect) . . . 8

1.4.6 Freescale (`metrowerks` Dialect) . . . 8

1.4.7 GNU C/C++ Compiler (GCC) (gnu Dialect) . . . 8

1.4.8 IAR Toolchain for 8051, ARM, AVR, MSP430, M16C and R8C (`iar-8051`, `iar-arm`, `iar-avr`, `iar-msp430` and `iar-m16c` Dialects) . . . 9

1.4.9 Keil Cx51 and C166 Optimizing C Compiler (`c51` and `c166` Dialects) . . . 9

1.4.10 Microsoft Visual C++ (`microsoft` Dialect) . . . 9

1.4.11 QNX QCC (`qnx` Dialect) . . . 9

1.4.12 Renesas H8S, H8/300 Series C/C++ Compiler (`renesas-h8` Dialect) . . . 9

1.4.13 Renesas RXC Toolchain (`renesas-rx` Dialect) . . . 9

1.4.14 TI Build Tools (`ti-armcl`, `ti-cl16x`, `ti-cl2000`, `ti-cl430`, `ti-cl470`, `ti-cl500` and `ti-cl55` Dialects) . . . 9

1.4.15 Wind River Diab Compiler (`diab` Dialect) . . . 10

1.5 Known Limitations . . . 10

2 Getting Started **11**

2.1 License Agreement . . . 11

2.2 Installing Goanna Studio . . . 11

2.2.1 Installing Goanna Studio Using Plugin Installation Wizard . . . 11

2.2.2 Installing Goanna Studio By Extracting Package Manually . . . 13

2.3 License Activation . . . 14

2.3.1 Online Activation . . . 14

2.3.2 Manual Activation . . . 14

2.3.3 Activating Floating License . . . 15

3 Using Goanna Studio **17**
3.1 Running Goanna on a Project . . . 17
3.2 Traces . . . 18
3.3 Stopping Goanna Analysis . . . 19
3.4 Goanna Project Properties Menu . . . 20
3.4.1 Toolchain Configuration . . . 20
3.4.2 Checks . . . 22
3.4.3 Files . . . 23
3.4.4 Options . . . 24
3.5 HTML Report of Analysis Results (Advanced) . . . 25
3.6 Analysis Results In XML File (Advanced) . . . 25
3.7 Goanna Global Preferences . . . 26
3.8 Checks Packages . . . 27
3.9 Using Goanna With PC-Lint Or FlexeLint (Advanced) . . . 28
3.10 Sample Code . . . 28

4 Getting the Best Results from Goanna **29**
4.1 Interprocedural Analysis . . . 29
4.2 A Word on False Positives . . . 29
4.3 Using the `_GOANNA` Preprocessor Symbol . . . 30
4.4 Using the `assert` macro . . . 30

5 Suppressing Warnings **31**
5.1 Suppressing Warnings . . . 31
5.2 Showing Suppressed Warnings . . . 31
5.3 Unsuppressing Warnings . . . 31
5.4 Warning Suppression Manager . . . 32
5.4.1 Toggling Suppression Status . . . 32
5.4.2 Searching and Sorting Warnings . . . 32
5.4.3 Other Features . . . 32

6 Goanna Dashboard **33**
6.1 Getting to the Goanna Dashboard . . . 33
6.2 Bug Statuses . . . 33
6.3 Severity . . . 33
6.4 Dashboard Views . . . 33
6.4.1 Project Page . . . 33
6.4.2 Report Page . . . 34
6.4.3 Directory Browser . . . 35
6.4.4 Warnings Browser . . . 36
6.4.5 Code Browser . . . 37
6.5 Database Upgrades . . . 38
6.6 Project Settings (Advanced) . . . 39
6.6.1 Code Browser Character Encodings . . . 39

7 Command Line Options **40**

7.1 Options 40

8 Support **44**

8.1 Known Issues 44

8.1.1 Known Issues For All Goanna Studio Eclipse Edition Users 44

8.1.2 Known Issues For QNX Momentics 44

8.1.3 Known Issues For Texas Instruments Code Composer Studio 45

8.1.4 Known Issues For Wind River Workbench 46

Index **47**

1 System Requirements

Before using Goanna, please check that your system and project meets the system requirements.

1.1 Operating Systems

1.1.1 Microsoft Windows

Goanna supports the following versions of Windows:

- Windows XP (Service Pack 2 or higher)
- Windows Vista
- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Windows Server 2003 (Service Pack 1 or higher)
- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2

Both 32-bit (x86) and 64-bit (x86-64/AMD64) versions of Windows are supported (except Windows XP and Windows Server 2003, which we only support 32-bit version).

Required Software

Before installing Goanna, you will need to install the following:

- Microsoft Visual C++ 2008 Redistributable (Download from http://www.microsoft.com/en-us/download/details.aspx?id=5582)

1.1.2 Linux

Goanna supports all major distributions of Linux with glibc (GNU C Library) 2.4 or higher installed.

Both 32-bit (x86) and 64-bit (x86-64/AMD64) versions of Linux are supported.

Using Goanna with SELinux enabled is not recommended.

1.1.3 Other Requirements

Some features of Goanna may require additional software or packages.

Requirements for Goanna Dashboard, HTML Report and Warning Suppression Manager

Goanna Dashboard (see 6), HTML Report (see 3.5) and Warning Suppression Manager (see 5.4) requires a web browser. The following web browsers are supported:

- Internet Explorer 9 or higher
- Microsoft Edge
- Mozilla Firefox - currently supported versions by Mozilla

- Google Chrome - currently supported versions by Google

While not officially supported, these features may also work on other modern browsers.

We also support Internet Explorer 7 and 8, however you may experience slow performance on these browsers; using Goanna Central with these browsers is not recommended.

Requirements for LM-X License Manager

Goanna uses LM-X License Manager 4.4.2 from X-Formation for licensing.

If you wish to use web-based UI of the License Manager (**Important:** Red Lizard Software does not provide full support for the web-based UI), the following software must be installed:

- A modern web browser
- Oracle Java Runtime Environment 1.6 or higher
- Adobe Flash Player

Please refer to X-Formation website (http://docs.x-formation.com/display/GEN/System+requirements+for+web-based+UIs) for more information.

Requirements for PC-Lint and FlexeLint

Goanna Studio has a facility to run PC-Lint and FlexeLint alongside Goanna analysis.

To use this facility, you need a valid installation of PC-Lint or FlexeLint, version 8.0 or higher.

1.2 IDE Requirements

Goanna Studio supports the following versions of Eclipse, and Eclipse-based IDEs:

- Eclipse 3.5 (Galileo) or higher with C/C++ Development Tools (CDT) 4.0 or higher.
  Note: If your installation of Eclipse does not have CDT installed, it will automatically be installed as a dependency while installing Goanna Studio.
- Analog Devices CrossCore Embedded Studio
- Atollic TrueSTUDIO 4.7.0 or higher
- Mentor Graphics Sourcery CodeBench
- QNX Momentics 3.1.0 or higher
- Texas Instruments Code Composer Studio 5.0 or higher
- Wind River Workbench 3.3 or higher

Important: Eclipse 3.4 (Ganymede) and older versions are not supported.

1.3 Hardware Requirements

Goanna requires, at the minimum, the following hardware:

- Processor: Intel Pentium 4 or higher
- Memory: 1 GB or more
- Storage: Minimum 1 GB of free disk space

For optimal analysis performance, we recommend at least the following:

- Processor: Intel Core 2 Duo or later CPU with minimum speed 2 GHz. Multi core CPUs are recommended.
- Memory: 4 GB or more
- Storage: 5 GB or more of free disk space

For large projects, Goanna may require more RAM and disk space than the ones shown here.

1.4 Supported Compilers

Goanna currently supports the following compilers:

Compiler Name	Goanna Dialect Name	Common Compiler Executables
Analog Devices C/C++ Compiler for SHARC Processors	`cc21k`	cc21k
ARM C/C++ Compiler	`armcc`	armcc, armlink
Cosmic Software C Cross Compiler	`cosmic`	cx6808, cx6812, cx6816, cxs12x, cxstm8
Freescale (formerly Metrowerks)	`metrowerks`	mwccarm, mwccmcf
GNU	`gnu`	gcc, g++, ld
IAR Toolchain for 8051	`iar-8051`	icc8051
IAR Toolchain for ARM	`iar-arm`	iccarm
IAR Toolchain for AVR	`iar-avr`	iccavr
IAR Toolchain for MSP430	`iar-msp430`	icc430
IAR Toolchain for M16C and R8C	`iar-m16c`	iccm16c
Keil Cx51 Optimizing C Compiler	`c51`	c51, cx51, bl51, lx51
Keil C166 Optimizing C Compiler	`c166`	c166, l166
Microsoft Visual C++	`microsoft`	cl, link
QNX QCC	`qnx`	qcc
Renesas H8S, H8/300 Series C/C++ Compiler	`renesas-h8`	ch38
Renesas RXC Toolchain	`renesas-rx`	ccrx
TI Build Tools - armcl	`ti-armcl`	armcl
TI Build Tools - CL16X	`ti-cl16x`	cl6x
TI Build Tools - CL2000	`ti-cl2000`	cl2000
TI Build Tools - MSP430	`ti-cl430`	cl430
TI Build Tools - CL470	`ti-cl470`	cl470
TI Build Tools - CL500	`ti-cl500`	cl500
TI Build Tools - CL55	`ti-cl55`	cl55
Wind River Diab Compiler	`diab`	dcc

Notes

- Green Hills compiler is no longer supported as of Goanna 3.1.0.
- As of Goanna 3.4.0, support for Tasking C166 compiler is only available upon request. Contact support@redlizards.com if you require support for this compiler.
- Goanna also ships with `cygwin`, `gnu-4.4.4` and `ti` dialects. However, these dialects exist only for backward compatibility. We strongly recommend that you do not use these dialects.

1.4.1 A Word On C99 and C++11 Support

Goanna strives to support most C99 and C++11 features as long as the compilers used in your projects also accept them. However, please note that, for C++11 extensions, the analysis engine generally does not make use of these extensions or any information derived from usage of these extensions. For example, Goanna does not perform any pointer or memory use related analysis on C++11 `std::shared_ptr`.

1.4.2 A Word On Compiler-Specific Syntax Extensions

Goanna strives to support most compiler-specific C/C++ syntax extensions for supported compilers. However, please note that even in cases where Goanna supports compiler-specific C/C++ syntax extensions, the analysis engine will generally not make use of these extensions or any information derived from usage of these extensions. For example, Goanna does not take Keil Cx51 Memory Models or Memory Types into account during analysis, even though Goanna supports relevant syntax extensions.

1.4.3 Analog Devices C/C++ Compiler for SHARC Processors (`cc21k` Dialect)

Goanna supports Analog Devices C/C++ Compiler for SHARC Processors shipped with Analog Devices CrossCore Embedded Studio 1.0 or higher.

1.4.4 ARM C/C++ Compiler (`armcc` Dialect)

Goanna supports the following versions of ARM C/C++ Compiler:

- RealView Development Suite (RVDS) versions 2.0 to 4.1 (inclusive)
- DS-5 Development Studio
- ARM Compiler versions 4.1 and 5.0 (including those shipped with Keil MDK-ARM versions 4 and 5)

RVDS 1.2 and older, ARM Developer Suite (ADS) versions 1.2 and older, and ARM Compiler 6 (`armclang`) are not supported.

Known Limitations

- Some compiler arguments, such as `--cpp` (for `--cpp` use `--c++` instead), `--kandr_include`, `--strict`, `--wchar`, `--no_wchar`, `--wchar16` and `--wchar32` are not supported. Goanna will ignore these arguments.
- Goanna will always include `RVCT<version>INC`, `ARMCC<version>INC` and `ARMINC` environment variables, and `<installation-path-of-compiler>\..\include` into the system include directories, even when `-J` option is used.

1.4.5 Cosmic Software C Cross Compiler (`cosmic` Dialect)

Goanna supports all recent versions of:

- CX6808 Compiler (part of Cosmic S08 and HC08 Cross Development Tools)
- CX6812 Compiler (part of Cosmic 68HC12 and HCS12 Cross Development Tools)
- CX6816 Compiler (part of Cosmic 68HC16 Cross Development Tools)
- CXS12X Compiler (part of Cosmic S12X and XGATE Cross Development Tools)
- CXSTM8 Compiler (part of Cosmic STM8 Cross Development Tools)

Note: CXXGATE Compiler (part of Cosmic S12X and XGATE Cross Development Tools) is not supported.

1.4.6 Freescale (`metrowerks` Dialect)

Goanna supports `mwccarm` compiler shipped with Freescale CodeWarrior Development Studio for Microcontrollers (CW MCU) version 10.2 or higher.

1.4.7 GNU C/C++ Compiler (GCC) (gnu Dialect)

Goanna supports most versions of GNU C/C++ Compiler (GCC).

Known Limitations

- `CPATH`, `C_INCLUDE_PATH` and `CPLUS_INCLUDE_PATH` environment variables are not recognized by Goanna.
- `--sysroot` option is not supported.
- Goanna ignores all optimisation options to work around known issues in recognising certain macros in GNU libc. Goanna may produce unexpected results if your source code relies on `__OPTIMIZE__`, `__OPTIMIZE_SIZE__`, or `__NO_INLINE__` predefined macros.

1.4.8 IAR Toolchain for 8051, ARM, AVR, MSP430, M16C and R8C (`iar-8051`, `iar-arm`, `iar-avr`, `iar-msp430` and `iar-m16c` Dialects)

Goanna supports IAR Toolchain for 8051, ARM, AVR, MSP430, M16C and R8C shipped with IAR Embedded Workbench 5.40 or higher.

Note: IAR Toolchain for AVR32 is not supported.

Known Limitations for IAR Toolchain for AVR compiler (`iar-avr` dialect)

- Goanna does not recognize QCCAVR environment variable.
- Goanna always recognize IAR syntax extensions. `-e` and `--strict` options, and `#pragma language` directive are ignored.
- C++ support for this compiler is limited. If your source code uses some standard library headers, such as `<vector>`, parse errors may occur. Additionally, `--ec++` or `--eec++` option causes Goanna to recognize full C++ language features, rather than Embedded C++ subset.

1.4.9 Keil Cx51 and C166 Optimizing C Compiler (`c51` and `c166` Dialects)

Goanna supports all recent versions of Keil Cx51 and C166 Optimizing C Compiler.

1.4.10 Microsoft Visual C++ (`microsoft` Dialect)

Goanna supports Microsoft Visual C++ compiler shipped with Microsoft Visual Studio 6.0 or higher.

Known Limitations

- Managed C++, C++/CLI and C++/CX syntax extensions, and CLR (Common Language Runtime) related compiler options are not supported.

1.4.11 QNX QCC (`qnx` Dialect)

Goanna supports QNX QCC shipped with QNX Momentics 4.7.0 or higher.

1.4.12 Renesas H8S, H8/300 Series C/C++ Compiler (`renesas-h8` Dialect)

Goanna supports `ch38` compiler shipped with Renesas C/C++ Compiler Package for H8SX, H8S, H8 Family (also called "H8S, H8/300 Series C/C++ Compiler") version 7.00 or higher.

1.4.13 Renesas RXC Toolchain (`renesas-rx` Dialect)

Goanna supports CC-RX compiler shipped with Renesas C/C++ Compiler Package for RX Family version 1.02 or higher.

1.4.14 TI Build Tools (`ti-armcl`, `ti-cl16x`, `ti-cl2000`, `ti-cl430`, `ti-cl470`, `ti-cl500` and `ti-cl55` Dialects)

Goanna supports the following Texas Instruments compilers:

- armcl compiler shipped with Texas Instruments Code Composer Studio versions 5 and 6,
- CL16X compiler shipped with Texas Instruments Code Composer Studio versions 2.2, 3 and 5,
  Note: CL16X compiler shipped with Texas Instruments Code Composer Studio version 4 is not supported.

- CL2000 compiler shipped with Texas Instruments Code Composer Studio versions 4 and 5,
- CL430 compiler shipped with Texas Instruments Code Composer Studio versions 4 and 5,
- CL470 compiler shipped with Texas Instruments Code Composer Studio version 5,
- CL500 compiler shipped with Texas Instruments Code Composer Studio version 5, and
- CL55 compiler shipped with Texas Instruments Code Composer Studio version 5.

Note: Except for armcl, compilers shipped with Texas Instruments Code Composer Studio version 6 are not supported.

Known Limitations

- Goanna recognizes compiler intrinsics for CL2000 compiler; however, they are not used for analysis.
- Goanna recognizes limited number of compiler intrinsics for armcl compiler; however, they are not used for analysis.
- Goanna does not recognize compiler intrinsics for any other Texas Instruments compiler; parse errors may occur if your source code uses them.
- Goanna is unable to auto-detect specific compilers in Texas Instruments Code Composer Studio. This means that, when you are using Goanna on Texas Instruments Code Composer Studio, you will need to manually select a compiler in the Toolchain Configuration (see 3.4.1).

  Important: If you do not select a compiler manually, Goanna will use a generic setting for TI Build Tools (`ti` dialect). However, this setting exists only for backward compatibility; using this is not recommended and you may encounter parse errors if you use the generic `ti` dialect.

1.4.15 Wind River Diab Compiler (`diab` Dialect)

Goanna supports Wind River Diab Compiler shipped with Wind River Workbench 3.3 or higher.

1.5 Known Limitations

In addition to those listed in the individual compiler support section, there are some known limitations that apply to some IDEs. See 8.1 for a list of these known limitations.

2 Getting Started

2.1 License Agreement

Before installing Goanna Studio, ensure you read the Goanna License Agreement, which can be found at

http://redlizards.com/license-term/evaluation-license-agreement/

if you are installing a trial version and

http://redlizards.com/license-term/

if you are installing a registered version of the tool. See Section 2.3 for how to purchase and install a license.

2.2 Installing Goanna Studio

There are two methods to install Goanna Studio on your Eclipse-based IDE depending on which IDE you use:

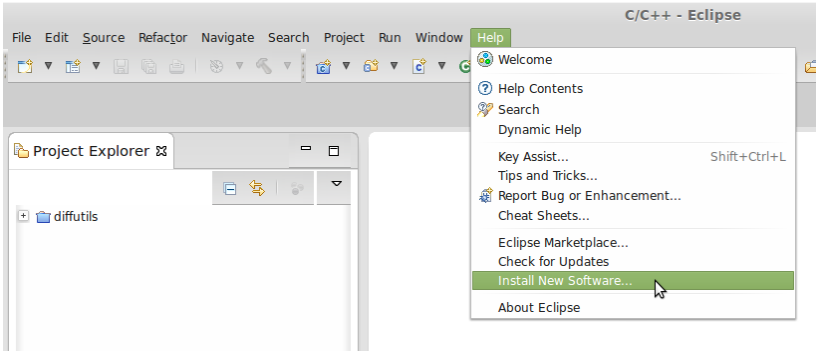
- If you use Eclipse, Analog Devices CrossCore Embedded Studio, Mentor Graphics Sourcery CodeBench, QNX Momentics, Texas Instruments Code Composer Studio, or Wind River Workbench, use the IDE's plugin installation wizard to install Goanna Studio. See 2.2.1 for the instructions.
- If you use Atollic TrueSTUDIO, you will need to install Goanna Studio by manually extracting Goanna Studio package into Atollic TrueSTUDIO installation directory: see 2.2.2 for the instructions.

2.2.1 Installing Goanna Studio Using Plugin Installation Wizard

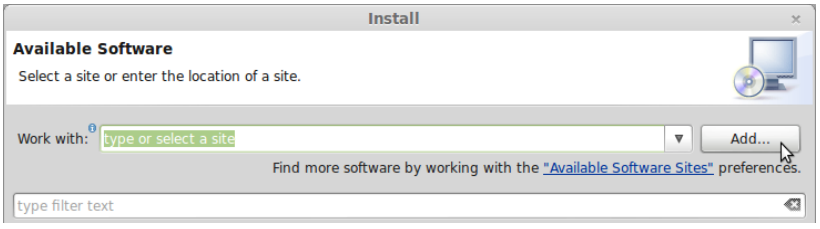
Important: If you are using Atollic TrueSTUDIO, you need to instead install Goanna Studio by manually extracting Goanna Studio package into Atollic TrueSTUDIO installation directory. See 2.2.2 for instructions on how to do this.

To install Goanna Studio using the plugin installation wizard, follow these steps:

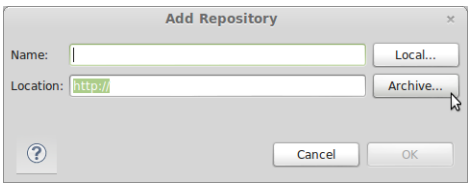
1. Place the distribution in your home directory. The package is typically named `goanna-eclipse-X.jar`, where X is the release number.
2. Start your IDE.
3. If you use Wind River Workbench, change the perspective to Advanced Device Development from Window > Open Perspective menu.
4. Go to Help > Install New Software.



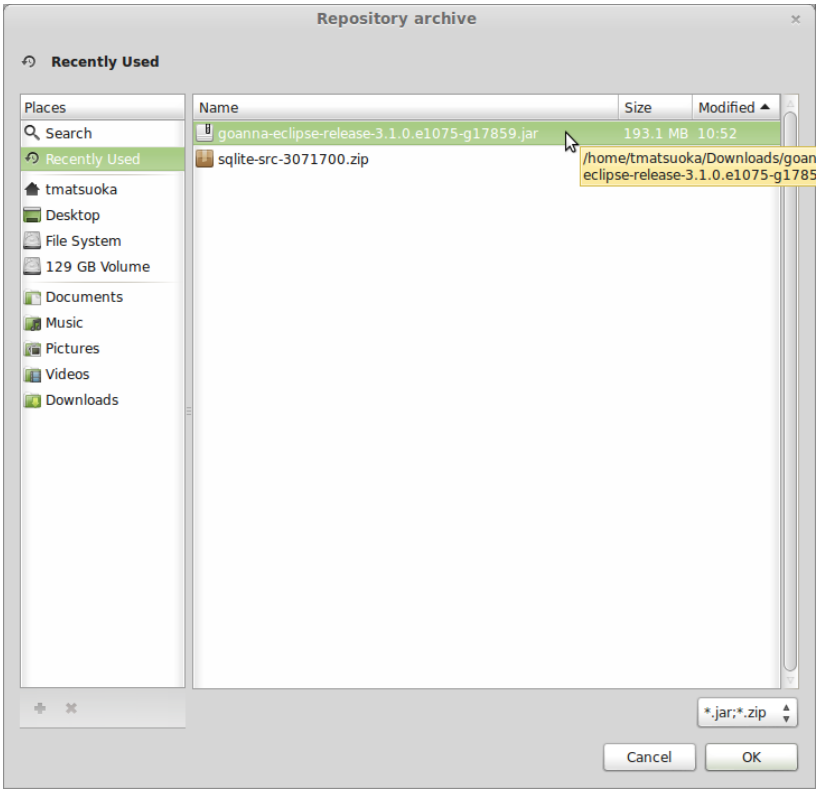
5. Click on Add.


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text

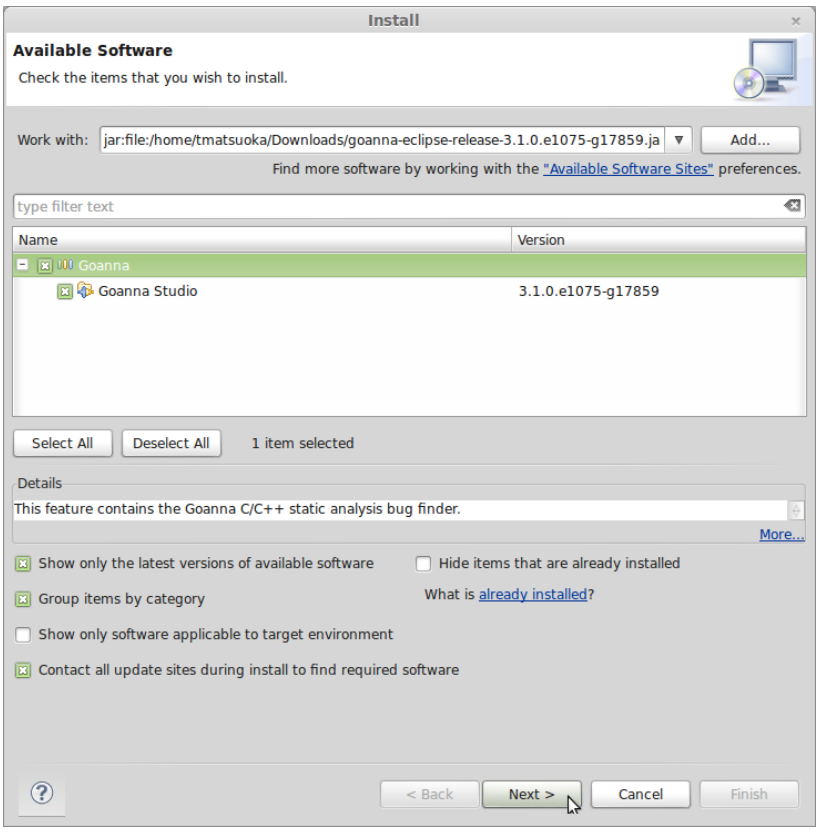
6. Choose to add from an Archive.



7. Select the Goanna Studio `jar` file.



8. Click OK.
9. Ensure the check box for Goanna Studio is checked and click Next.



10. Click Next.

From this point on, the wizard will direct you through the installation. It will ask you to accept the license agreement, and depending on your current configuration, it may install components that are required but missing, such as CDT.

After installation you will have to restart your IDE to use Goanna Studio.

2.2.2 Installing Goanna Studio By Extracting Package Manually

Important: If you are *not* using Atollic TrueSTUDIO, you should instead use the plugin installation wizard to install Goanna Studio. See 2.2.1 for instructions on how to do this.

If you are using Atollic TrueSTUDIO, you will need to install Goanna Studio by extracting the package manually into the installation directory of Atollic TrueSTUDIO. To do so, follow these steps:

1. Exit all running instances of Atollic TrueSTUDIO.
2. Rename the file extension of Goanna Studio package file (with file extension `.jar`) to `.zip`.
3. Right-click the renamed package, and click Extract All... to extract the package.
   Alternatively, you can also use any ZIP archive extractor to extract the package.
4. Inside the extracted package, you should find `features` and `plugins` folders.
5. Move all files and directories inside `features` folder to:
   `<installation-directory-of-atollic-truestudio>\ide\features`.
6. Move all files and directories inside `plugins` folder to:
   `<installation-directory-of-atollic-truestudio>\ide\plugins`.
7. Start Atollic TrueSTUDIO.

2.3 License Activation

Whether you are just evaluating Goanna or have purchased the full version, you must activate your license before you can use Goanna.

You should have received an email containing your license information from Red Lizard Software. This email will contain an Order Number that you will need below.

Then start the activation process by opening Preferences in Goanna menu.

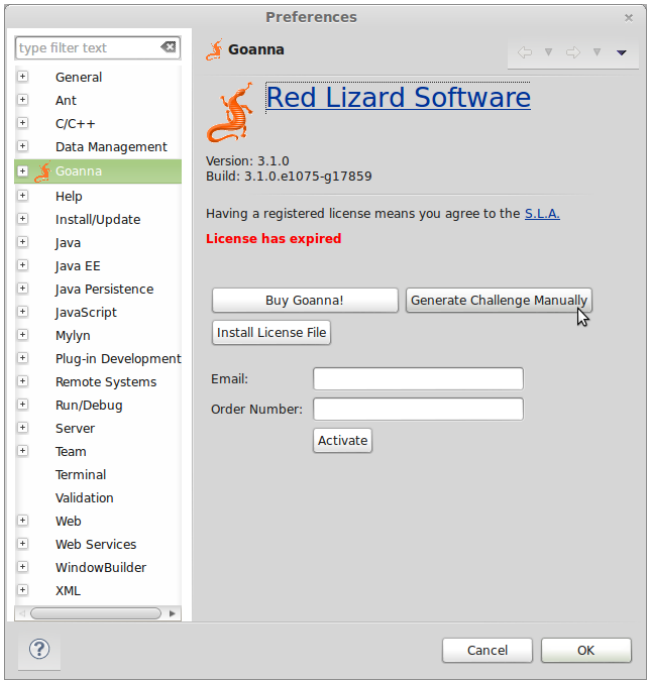
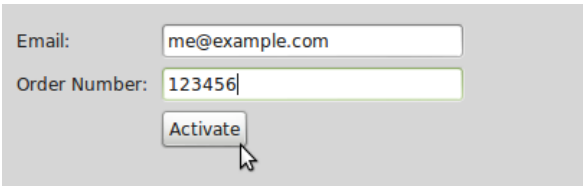


Figure 1: Activation screen

2.3.1 Online Activation

At this point, you can now activate your license over the Internet connection. Follow these steps to complete the activation:

1. Type your e-mail address and the order number you received into Email and Order Number textboxes.

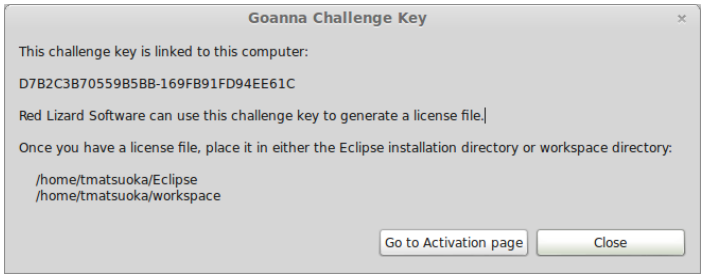


2. Press Activate to finish activation.

2.3.2 Manual Activation

Alternatively, you can also manually activate your license by obtaining license file at the Red Lizard Software website first. To do so, follow these steps:

1. Click Generate Challenge Manually at the activation screen.
2. The screen will then show the challenge key of your machine.


Goanna Challenge Key
This challenge key is linked to this computer:
D7B2C3B70559B5BB-169FB91FD94EE61C
Red Lizard Software can use this challenge key to generate a license file.
Once you have a license file, place it in either the Eclipse installation directory or workspace directory:
/home/tmatsuoka/Eclipse
/home/tmatsuoka/workspace
Go to Activation page
Close

3. Go to http://www.redlizards.com/purchase/activate-license/.
4. Type in your e-mail address and the order number.
5. Copy the challenge key into Challenge text box.
6. Read and agree the license agreement, then press Activate.

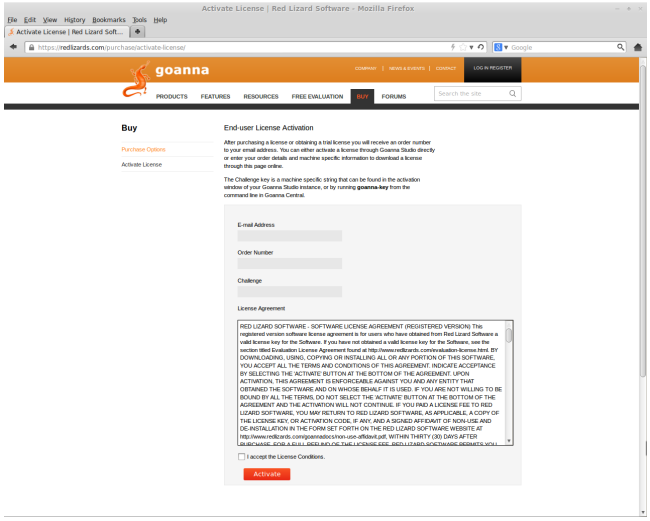


Figure 2: Activation web page

7. Download your license file via the generated download link.
8. Click Install License File at the activation screen, and choose the license file you have downloaded in the activation screen to finish activation.

2.3.3 Activating Floating License

If you have floating (network) license, follow these steps to borrow a seat from the license server:

1. Expand Goanna preferences entry on the left hand pane, then click Settings.
2. Type the host name, and optionally port number, of your license server into License textbox.
3. Enter the borrow duration into Borrow Duration textbox.
4. Press OK to finish activation.

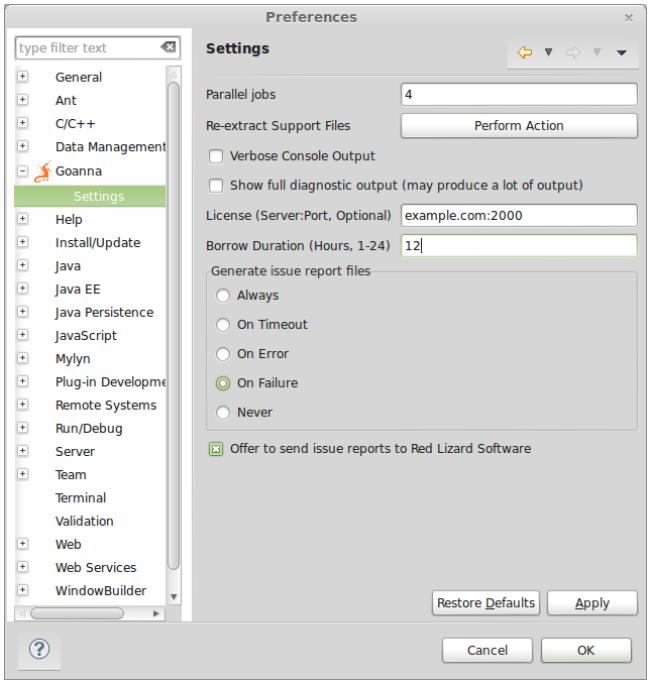


Figure 3: Activating floating license

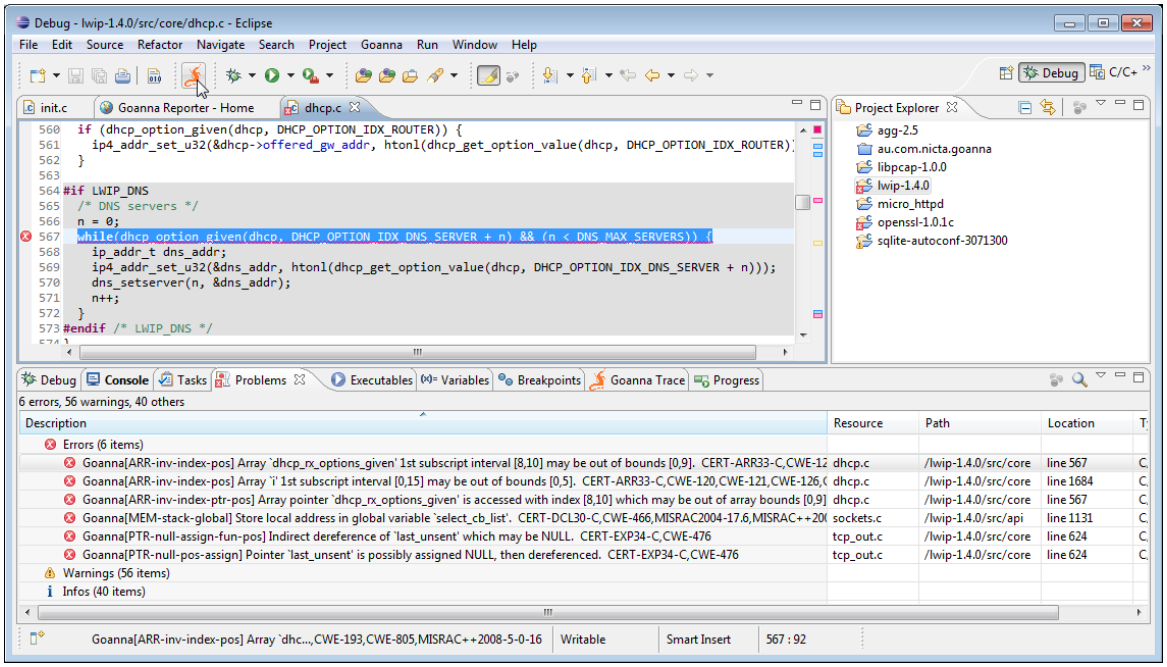
3 Using Goanna Studio

3.1 Running Goanna on a Project

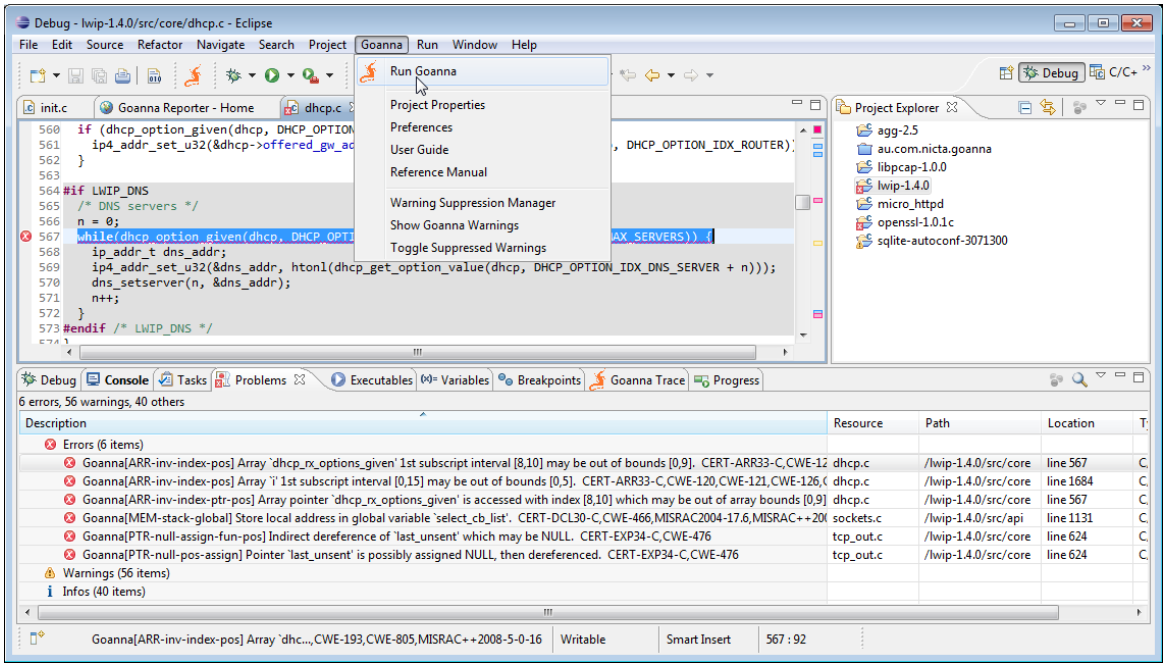
Important: If you are using Texas Instruments Code Composer Studio, you should configure the toolchain setting to select the compiler used by the project before running Goanna analysis. See 3.4.1 for instructions on how to do this.

Quick Solution Goanna Studio will appear as an icon on the toolbar. To run Goanna Studio over your code, select a project and click on the icon. Goanna Studio will automatically display the warnings, if there are any. □

Goanna Studio can be invoked via a button with the Goanna icon, located on the toolbar.



To run Goanna Studio you will first need to select a project. Goanna Studio acts as a compiler and reports bugs in a similar fashion to a compiler.



It can also be invoked via the Goanna menu as shown above:

Run Goanna : Run Goanna on the selected project or files.

Any results from Goanna Studio will appear in the Eclipse Problems view. The description of each warning takes the following format:

```
Goanna[SPC-return] Missing return statement on some paths. CERT-MSC37-C
    CWE-758 MISRAC2004-16.8 MISRAC++2008-8-4-3
```

The description consists of several parts:

- Goanna to distinguish Goanna warnings from other sources of markers in the Problems view.
- The *check name* (for example, `[SPC-return]`. This is the unique identifier of the Goanna check, used everywhere to identify the check, such as in the Reference Guide, the Describe Check help information, and in the Checks tab of Goanna's project/solution settings.
- The text of the warning message itself, for example, `Missing return statement on some paths`.
- A list of rule(s) from coding standards to which the warning applies, for example, `CERT-MSC37-C CWE-758 MISRAC2004-16.8 MISRAC++2008-8-4-3`. This cross references Goanna warnings back to the corresponding rule(s) in coding standards, whether or not you have enabled all the checks in those standards.

Each warning also identifies the file and line number, in much the same way as the compiler's warning and error messages. Double-clicking a message will place the cursor at the relevant part of the code.

3.2 Traces

Many potential defects only occur on particular execution paths through your program. This execution path is called a *trace*.

For each warning, Goanna Studio can show you the trace in the Goanna Traces tab, allowing you to step through the exact sequence of statements leading to the potential defect.

To view the trace for a warning, right-click on it in the Goanna Warnings tab, and select Show Trace from the context menu.

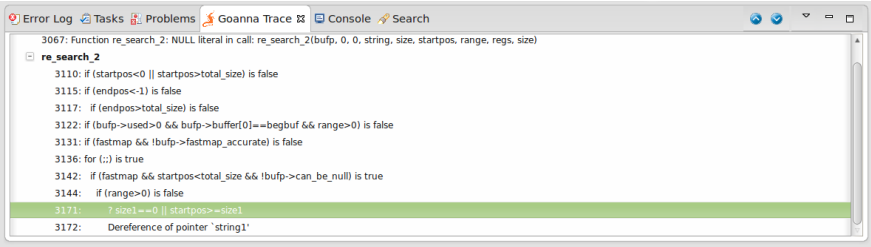


Figure 4: Trace view.

Each row in the Goanna Trace tab is a sequence point in your program. You can click on a row, use the Back and Next buttons on the top right, or press Up or Down arrow keys to highlight the line of code for each sequence point in the trace. Where the control flow passes through a conditional or looping statement such as `if`, `for`, or `while`, the Description will show:

- `if <condition> is false`
- `if <condition> is true`

to indicate which branch is taken. In addition, each significant event leading up to the warning, such as the call to `malloc` in a potential memory leak (`MEM-free-some`) defect, is shown in the Description column.

If the control flow passes through function boundary, the name of the function (and additionally file name if different from previous) will be shown in bold text to indicate this. You can collapse or expand sequence points in the function.

3.3 Stopping Goanna Analysis

To stop Goanna analysis while it is currently running, open Goanna menu and click Stop Goanna.

3.4 Goanna Project Properties Menu

Under the Goanna Project Properties menu, there are five major submenus which we detail hereafter.

3.4.1 Toolchain Configuration

The Compiler tab displays current toolchain settings, and optionally change them.

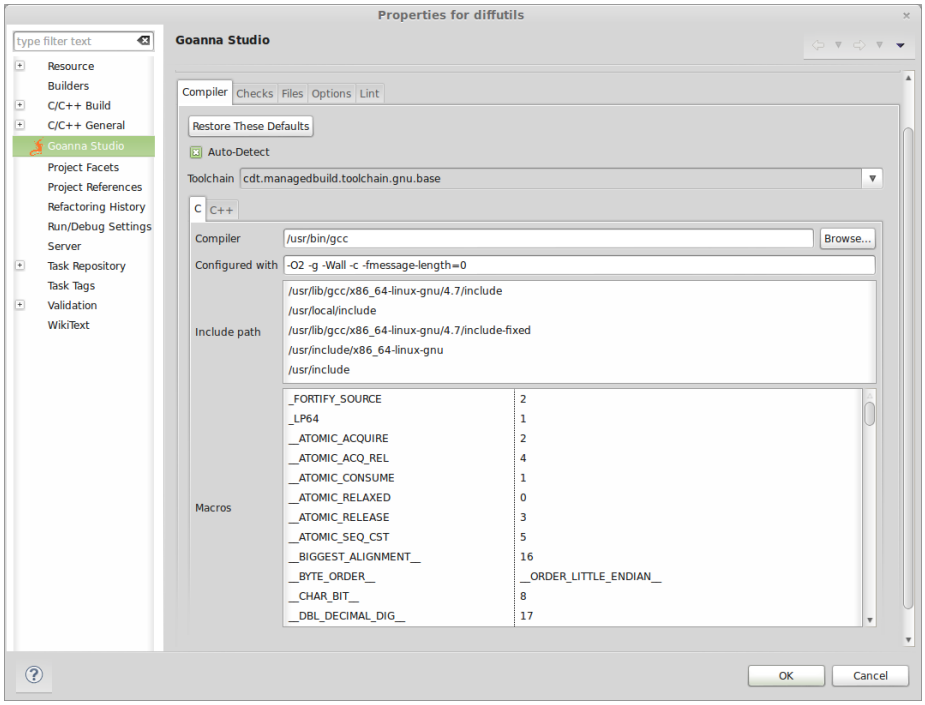


Figure 5: Compiler tab

Automatically Setting Up Toolchains

By default, the Auto-Detect option is enabled, which means that Goanna will try to automatically detect toolchain settings for known compilers based on information available from the project settings. Settings detected automatically by Goanna are displayed in Compiler, Configured With, Include path and Macros.

Manually Choosing Toolchains

There may be cases where you may wish to change the toolchain settings:

- Necessary information for toolchain automatic detection was unavailable, or
- You wish to use different toolchain settings from what Goanna has detected.

The Toolchain option allows you to select a toolchain from the list of configurations bundled with Goanna. The list of available toolchains are as follows:

Compiler Name	Goanna Dialect Name	Common Compiler Executables
Analog Devices C/C++ Compiler for SHARC Processors	`cc21k`	cc21k
ARM C/C++ Compiler	`armcc`	armcc, armlink
Cosmic Software C Cross Compiler	`cosmic`	cx6808, cx6812, cx6816, cxs12x, cxstm8
Freescale (formerly Metrowerks)	`metrowerks`	mwccarm, mwccmcf
GNU	`gnu`	gcc, g++, ld
IAR Toolchain for 8051	`iar-8051`	icc8051
IAR Toolchain for ARM	`iar-arm`	iccarm
IAR Toolchain for AVR	`iar-avr`	iccavr
IAR Toolchain for MSP430	`iar-msp430`	icc430
IAR Toolchain for M16C and R8C	`iar-m16c`	iccm16c
Keil Cx51 Optimizing C Compiler	`c51`	c51, cx51, bl51, lx51
Keil C166 Optimizing C Compiler	`c166`	c166, l166
Microsoft Visual C++	`microsoft`	cl, link
QNX QCC	`qnx`	qcc
Renesas H8S, H8/300 Series C/C++ Compiler	`renesas-h8`	ch38
Renesas RXC Toolchain	`renesas-rx`	ccrx
TI Build Tools - armcl	`ti-armcl`	armcl
TI Build Tools - CL16X	`ti-cl16x`	cl6x
TI Build Tools - CL2000	`ti-cl2000`	cl2000
TI Build Tools - MSP430	`ti-cl430`	cl430
TI Build Tools - CL470	`ti-cl470`	cl470
TI Build Tools - CL500	`ti-cl500`	cl500
TI Build Tools - CL55	`ti-cl55`	cl55
Wind River Diab Compiler	`diab`	dcc

Note: Goanna also ships with `cygwin`, `gnu-4.4.4` and `ti` dialects. However these dialects exist only for backward compatibility. We strongly recommend that you do not use these dialects.

Manually Setting Up Compiler Location

If required, it is also possible to select the location of your compiler and the arguments to be passed. To do so, select compiler executable in Compiler option, and fill in required arguments in Configured with box.

Note: Goanna currently supports only the compilers listed above; manually selected compilers still need to be one of these for Goanna to recognize.

3.4.2 Checks

The Checks tab lets you enable and disable the checks that Goanna performs.

To configure and select the checks that Goanna Studio performs in each run, you can select one of three options:

- Default: This selects the Goanna Core set of checks, which produce the most accurate and complete diagnostics for most projects. This option is recommended for most users.
- Coding Standards: This allows you, in addition to Goanna Core, to optionally include checks from a number of industry coding standards. This option is useful for checking compliance of your code with these standards. In addition to the default set of checks (Goanna Core), Goanna also supports the following standards:

Standard code	Standard name
`cert`	Computer Emergency Response Team (CERT) C/C++ Coding Standard
`cwe`	Common Weakness Enumeration (CWE)
`misrac2004`	Motor Industry Software Reliability Association (MISRA) C:2004
`misrac++2008`	Motor Industry Software Reliability Association (MISRA) C++:2008
`misrac2012`	Motor Industry Software Reliability Association (MISRA) C:2012

- Custom: This allows you enable or disable each check individually. Click on the checkboxes to toggle each check or group of checks. You can also use the Select All and Select None buttons when the Custom option is selected.

For a description of a specific check simply click on the check code and short description next to the checkbox. Details and examples of the checks can be found in the companion reference manual *Goanna Reference Manual*.

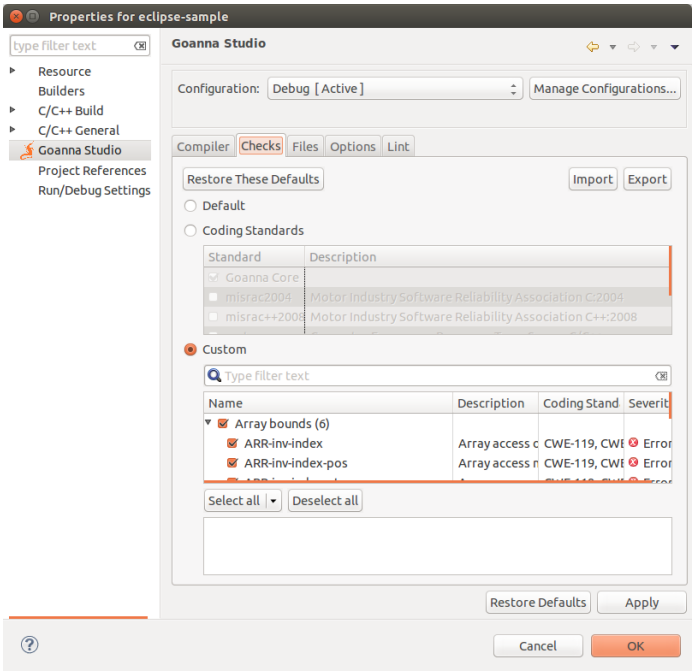


Figure 6: Goanna Settings for Checks

Checks can be filtered by using the Filter option, simply type in what you would like to filter by and the checks will be updated. Deleting any text in the Filter text box will display all the checks. Note that the Select All and Select None buttons apply to the filtered checks, easily allowing you to add remove checks by filtering.

For example, in Custom mode you can easily enable all the Common Weakness Enumeration (CWE) checks by typing cwe in the Filter box, and clicking Select All. Then clear the Filter box again to see the effect.

Exporting And Importing Check Selection

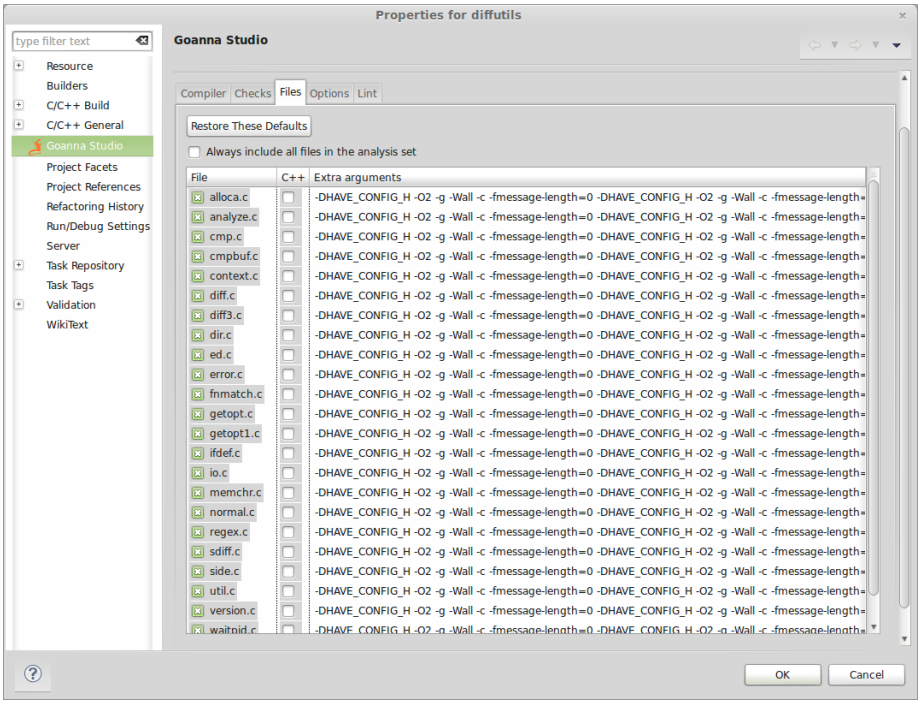
Goanna Studio also supports exporting and importing the check selection status so that you can easily configure multiple machines to use the same set of checks.

To export checks selection to a file, click the Export button and specify the location of the file to be saved. Once the selection is exported to a file, you can then import this setting by clicking the Import button.

Note that, the exported checks selection also contains information about what checks packages are enabled, to ensure all necessary checks are available. This means that, for example, if you copy checks selection to a different machine with different checks packages enabled, then this may result in enabling more checks than intended. Goanna Studio will warn at import time in such cases.

3.4.3 Files

To alter which files are analyzed and how they are analyzed, right click on a project and select the Properties menu item. Within the Goanna Studio section there five tabs. Select the Files tab.



By default all files in the project will be analysed. If you wish to exclude some files, uncheck the boxes next to their file names.

Checking Always include all files in the analysis set will force Goanna to analyse all files in the project.

If you select CXX checkbox, it will instruct Goanna to analyse these files as C++ files. Otherwise Goanna will assume they are C files.

Clicking Extra Arguments column will allow you to change any arguments passed to Goanna.

3.4.4 Options

The Options tab contains advanced options that control how Goanna analyzes your code.

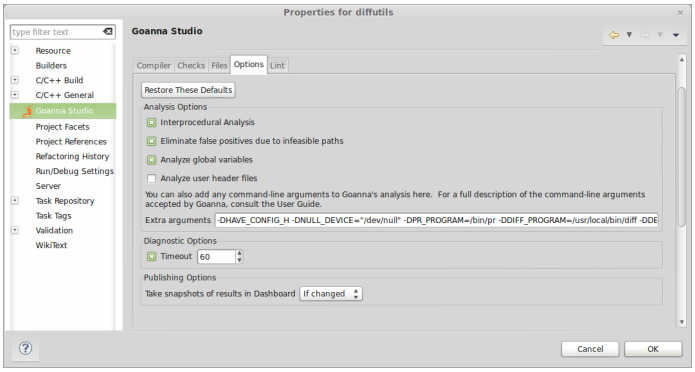


Figure 7: Goanna Options

A description of these Goanna Studio options is as follows:

- Interprocedural analysis (default: yes): This enables interprocedural analysis. See section 4.1 later in this document for a detailed description of this feature.
- Eliminate false positives due to infeasible paths (default: yes): This enables false positive elimination based on a Satisfiability Modulo Theories (SMT) solver algorithm. See section 4.2 later in this document for a detailed description of this feature.
- Analyze global variables (default: yes): Instructs Goanna to include global variables in its analysis.
- Analyze user header files (default: no): Instructs Goanna to include user headers (generally, those included using `#include "..."` syntax) into analysis.
- Extra arguments (default: none): This allows you to pass arguments to the Goanna executable. Most users will not need to use this field, because Goanna Studio synthesizes the correct arguments from project files.

  In some cases, though, a user may wish to pass arguments not found in a project file. When running Goanna, you may want to search for include files in particular directories using the -I flag: `-I C:\SomeIncludePath`. A list of the available arguments to the Goanna executable is detailed in section 7.1.
- Analyse the source code with this working directory (default: `'.'`): This option allows to change the directory where Goanna analysis runs from.
- Timeout (default: 240 seconds): This is the maximum number of seconds Goanna is allowed to spend in each analysis phase within one source file.

  Generally speaking, increasing timeout may result in more accurate results, but will take longer to complete the analysis. Decreasing timeout will improve the running time, but may result in less accurate results. Due to the underlying technology of the Goanna analysis engine, this timeout is essential.

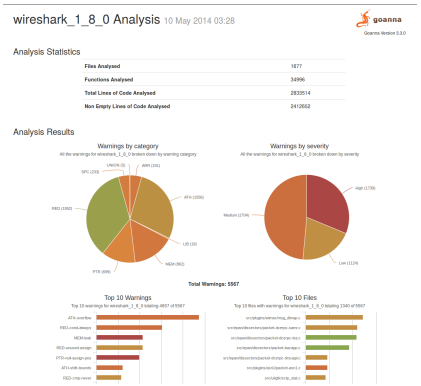
  Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!
- Automatically add snapshot at each Goanna run (default: Always): By default Goanna automatically generates a new snapshot for the Dashboard (see 6) after each analysis run. You can change this to Never to never generate a snapshot automatically.

3.5 HTML Report of Analysis Results (Advanced)

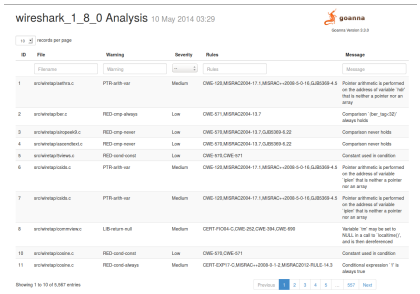
Goanna can generate an HTML report of the analysis results. There are two types of HTML reports:

- *Summary Report*, which shows a summary of the analysis result, including:
  - Basic statistics, such as the number of files analyzed,
  - Per-category and per-severity warnings pie chart,
  - Bar charts of the top 10 warnings per check and top 10 files with warnings, and
  - List of warning numbers per check.

  This report shows the high-level statistics of the analysis results in an easy-to-see format; and
- *Warnings Report*, which shows a table of all warnings found; this table is interactive and supports filtering.



Example of Summary Report



Example of Warnings Report

To generate HTML report files at the end of analysis, add **`--html-report`** option to Extra arguments field in the Options tab of the Project Properties. This option accepts an optional argument to specify the type(s) of report to be generated:

- **`--html-report=summary`**: Generates summary report file.
- **`--html-report=warnings`**: Generates warnings report file.
- **`--html-report=all`** or **`--html-report`** with no extra argument: Generates both summary and warnings report files.
- **No `--html-report` option**: HTML report files will not be generated.

Generated HTML report files will be saved to the directory where your project is located.

3.6 Analysis Results In XML File (Advanced)

Goanna can optionally output the analysis result to a XML file. This is useful if you need Goanna to be used in conjunction with some other platform or framework and need programmatic access to the analysis result.

To generate a XML output file at the end of analysis, add **`--output-xml=<xml-file-name>`** option to Extra arguments field in the Options tab of the Project Properties.

Generated XML output file will be saved to the specified file name in the directory where your project(s) are located.

Note: If the specified XML output file already exists, Goanna will overwrite the file at the end of next analysis.

3.7 Goanna Global Preferences

In addition to per-project settings, Goanna also has some global settings options. To access these options, either:

- Click Goanna Preferences in Goanna menu, then expand the Goanna entry on the left and navigate to Settings, or
- Open Eclipse preferences, then select Goanna> Settings.

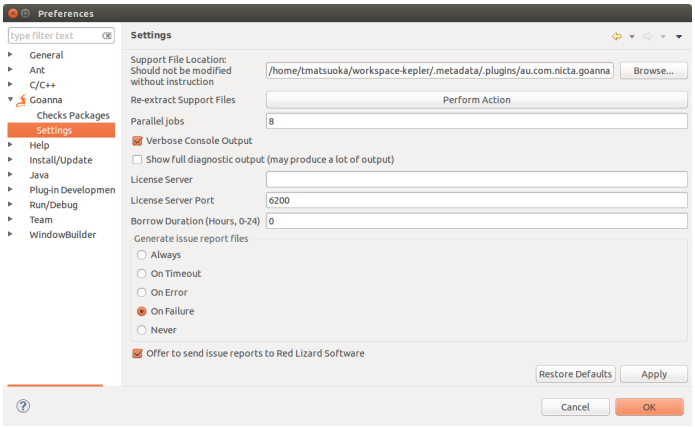


Figure 9: Goanna Global Preferences Window

A description of these Goanna Studio options is as follows:

- Support File Location **(Diagnostics Only)**: **Do not modify this option unless directed by Red Lizard Software support team.**
- Parallel Jobs: This option specifies how many (maximum) Goanna instances should run at one time (from 1 core up to 20 cores). By default this is set to the number of CPU cores available on the system.
- Re-extract Support Files: Extracts Goanna support files into Eclipse workspace.
  Important: Usually this is performed when Goanna Studio is installed, so there is no need to use this option. Only use this to repair the current installation.
- Verbose output: Enables verbose output mode, which causes Goanna to display command line arguments and Goanna output from the console.
- Show full diagnostics output: Enables even more verbose output, which includes full Goanna debugging output.
  Important: You should not enable this option unless you experience Goanna failures or crashes. Enabling this option will also slow down analysis and produce a lot of output.
- License: If you use Goanna with floating license (see 2.3.3), type the address and port number of the license server here.
- Borrow Duration: If you use Goanna with floating license (see 2.3.3), type the borrowing hours here.
- Generate issue report files : Goanna can automatically generate issue reports when you experience Goanna failures or errors, which can then be sent to Red Lizard Software to assist diagnosis. You can choose when to generate issue reports from the following options:
  - Never: Never generate issue reports.
  - On failure (default for non-evaluation users): Generate issue reports when Goanna experiences fatal error.
  - On error (default for evaluation users): Generate issue reports when Goanna experiences fatal error or PARSE error.
  - On timeout: Generate issue reports when Goanna experiences fatal error, PARSE error, or when analysis times out.
  - Always **(Not Recommended)**: Always generate issue reports after every analysis, *even if successful*.
- Offer to send issue reports to Red Lizard Software: If this option is checked, then when Goanna generates issue reports they will be displayed in a window which allows you to send the issue reports to Red Lizard Software for diagnosis.
  Important Note: Issue reports contain your name or e-mail address (extracted from license file), and additionally some information extracted from source files themselves. Please examine the content of the report carefully before submitting them to Red Lizard Software.

3.8 Checks Packages

Goanna Studio comes with a default package of checks that you can choose to enable via the Checks tab as described in Section 3.4.2. It is also possible to install more checks, specifically for the purpose of providing extra conformance checking for selected C/C++ standards.

Goanna ships with the following checks packages:

- **`stdchecks`** (enabled by default): "Goanna Core" checks. This checks package contains a set of checks for common C/C++ issues.
- **`security`**: Checks that are targetting well known security vulnerabilities.
- **`misrac2004`**: Dedicated checks for MISRA-C:2004 coding standard.
- **`misrac++2008`**: Dedicated checks for MISRA C++:2008 coding standard.
- **`misrac2012`**: Dedicated checks for MISRA C:2012 coding standard.

It is also possible to add custom checks packages to the analysis results. Contact Red Lizard Software for more information about custom checks packages.

To select which packages are installed, either:

- Click Goanna Preferences in Goanna menu, then expand the Goanna entry on the left and navigate to Checks Packages, or
- Open Eclipse preferences, then select Goanna> Checks Packages.

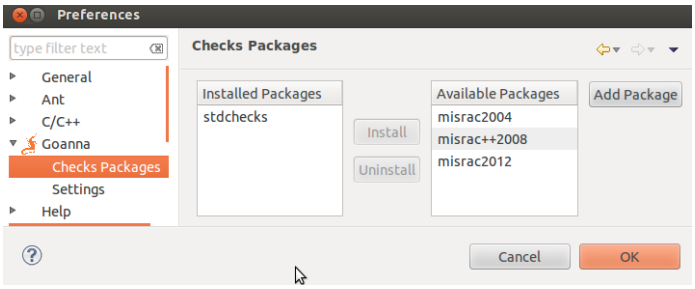


Figure 10: Goanna Checks Package Preferences

To enable a checks package, click the checks package you wish to enable from Available Packages list, then click Install. To disable a checks package, click the checks package you wish to disable from Installed Packages list, then click Uninstall.

If you have a custom checks package, then you can add it by clicking Add Package.

Important Note

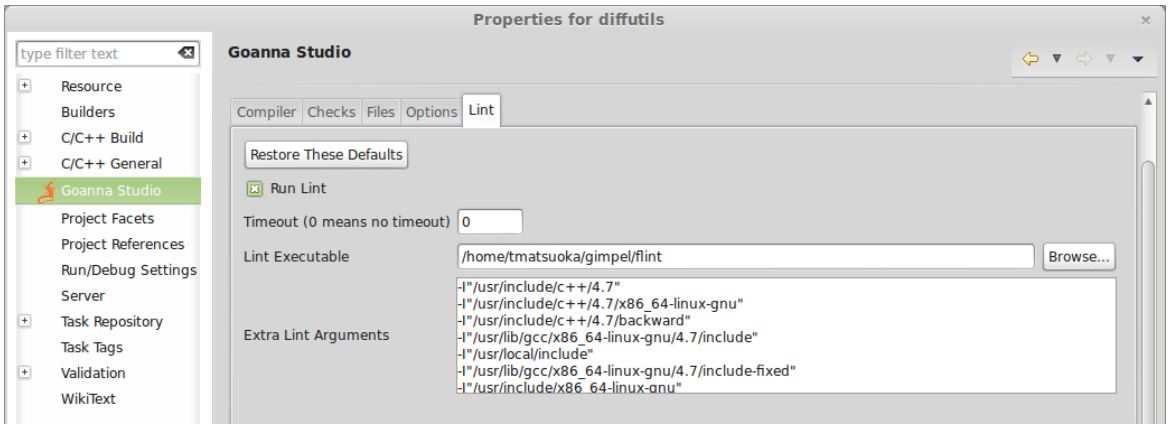
Checks package operations are global and may affect analysis on all existing *and new* projects. Additionally, enabling a checks package will also change the default set of checks to be enabled for all existing *and new* projects.

Installing or uninstalling checks packages, and adding custom checks packages require write access to your workspace directory.

3.9 Using Goanna With PC-Lint Or FlexeLint (Advanced)

The Eclipse edition of Goanna Studio has a facility to run PC-Lint or FlexeLint at the same time as Goanna analysis. To set up the project to run PC-Lint or FlexeLint alongside Goanna, follow these steps:

1. Go to Projects Properties page and choose the Goanna Studio section. Select the Lint Tab.



2. Specify the location of the Lint executable in the second text box.
3. Tick the Run Lint check-box.
4. If you wish to pass any additional arguments to Lint, use the third text box. By default it has the options for including the compiler's standard header files. Please refer to your Lint manual for a specification of the Lint options you can use.

   A `lnt` directory is generated if it does not already exist. Inside is placed a `goanna.lnt` file which modifies Lint's output such that it can be processed by Goanna Studio. A second file, `source_files.lnt`, is placed in this directory. This file lists all the C/C++ source files in your project.

   From now on, when you analyze a project with Goanna, the Lint executable that you specified will be run with the additional arguments you provided, as well as the two generated `.lnt` files. Once these files are generated, they will not be replaced, so it is possible to modify them for subsequent analysis runs.

   Any Lint warnings will now appear alongside Goanna's warnings, in the Goanna Warnings tab. Right-clicking on one of these warnings and selecting Trace will allow you to view the trace of the warning in the same way that you can with Goanna's warnings.

3.10 Sample Code

Important: This sample code package can only be used with Eclipse with GNU C/C++ Compiler (`gcc`); it is not compatible with any other IDEs or compilers.

A package containing a number of sample C/C++ files is available on our website. Go to http://www.redlizards.com/resources/example-code/ and download the Eclipse Sample Code package. The files in this package may be useful for practicing using Goanna, or ensuring that it is working correctly. To start using the sample code, proceed as follows:

1. launch Eclipse.
2. Create a new C++ project in Eclipse, by selecting File>New>C++ Project. Give it a name, select Linux GCC from the Toolchains box, and click Finish.
3. Right-click the new project in the Project Explorer window, and click Import. Select Archive File under the General heading, and click Next.
4. Click Browse, find the archive `eclipse-test.zip` in the `sample-files` directory, and select it. Then, click Finish.

Once the files are imported, you can now try analyzing the project with Goanna Studio.

4 Getting the Best Results from Goanna

4.1 Interprocedural Analysis

Goanna's interprocedural analysis propagates information about function behaviour to other functions. This information includes parameter values, return values, and function effects that may impact other parts of the code. This enables Goanna to detect things in your program such as freeing of memory through function calls, functions that never return, and input values to some functions.

Interprocedural analysis is not limited to a specific set of checks, but rather enhances the precision of many checks.

An example of what interprocedural analysis can find can be seen in the sample of function `myAlloc`.

```
void *myAlloc(int param){
  void *p = malloc(param);
  if (p)
     return p;
   else
     return NULL;
}

int main(int argc, char ** argv) {
  int * n;
  n = (int*)myAlloc(sizeof(int) * 10);
  n[0] = 5; // this may be a dereference of NULL
  return *n;
}
```

Here, Goanna learns that `myAlloc` may return `NULL`. This means that when the return value of `myAlloc` is assigned to `n`, Goanna knows this value may be `NULL`. Therefore, the expression `n` may be dereferencing a `NULL` pointer, and Goanna will warn accordingly.

There is some additional computation overhead in running interprocedural analysis. If you need rapid results without much depth, then turning off interprocedural analysis will provide faster results (at the cost of accuracy in some checks). To turn off interprocedural analysis, use the `--no-ipa` option.

By default interprocedural analysis does two passes (in optimized order) over each file. This provides a good approximation for function behaviours, but may miss some complex behaviours that require many passes to accurately detect. Additional precision can be gained by increasing the iteration limit (the maximum number of passes Goanna will do). To change the interprocedural analysis iteration limit, use the `--ipa-iterations` option.

4.2 A Word on False Positives

Goanna considers all possible execution paths in your program, and will warn you if it finds potential defects (such as use of an uninitialised variable) that occur only on particular execution paths and not others. But sometimes, the execution path leading to a potential defect is actually not possible when the program is executed. If Goanna is able to deduce this through static analysis, then it won't warn you. But if it can't, then you may receive a spurious warning for a defect that isn't really there. Such warnings are called *false positives*.

Some false positives occur because Goanna currently does not track dependencies between variables in loops. For example, if you have a loop with two counters and only test one:

```
char buffer[11];
int i, count;

i = 0;
count = 10;
while (--count >= 0) {
  buffer[i++] = 'x';
}
buffer[i] = '\0';
```

Goanna may issue a false positive warning because it doesn't deduce that `i = 10` when the loop terminates.

Such false positives can often be suppressed with the `assert` macro (sec. 4.4). Otherwise you can suppress false positives using the Goanna Dashboard (see 6.2).

4.3 Using the _GOANNA Preprocessor Symbol

Goanna has a built-in preprocessor definition, defined by the macro

```
#define _GOANNA 1
```

This allows code to be explicitly included in or excluded from analysis by Goanna. For example:

```
#ifdef _GOANNA
//Code only to be included while the program is being analysed
#endif
#ifndef _GOANNA
//Code not to be analyzed by Goanna
#endif
```

4.4 Using the `assert` macro

Goanna can sometimes use information provided by `assert()` to refine its analysis of numerical and pointer values. It does this by using `assert` statements as assumptions for value ranges and pointer validity.

For example, in the code below:

```
  void my_fun(void) {
    int my_array[20];
    int x = rand();

    assert(x == 10);
    f(my_array[x]);
  }
```

the `assert()` means that the array reference must be in-bounds, even though the index variable x has a randomly-assigned value. Therefore, Goanna does not issue an out-of-bounds warning.

5 Suppressing Warnings

For a number of reasons, you may wish to ignore specific warning messages without modifying your check settings. Goanna Studio allows you to easily hide and show warnings to help keep emphasis on the most critical bugs.

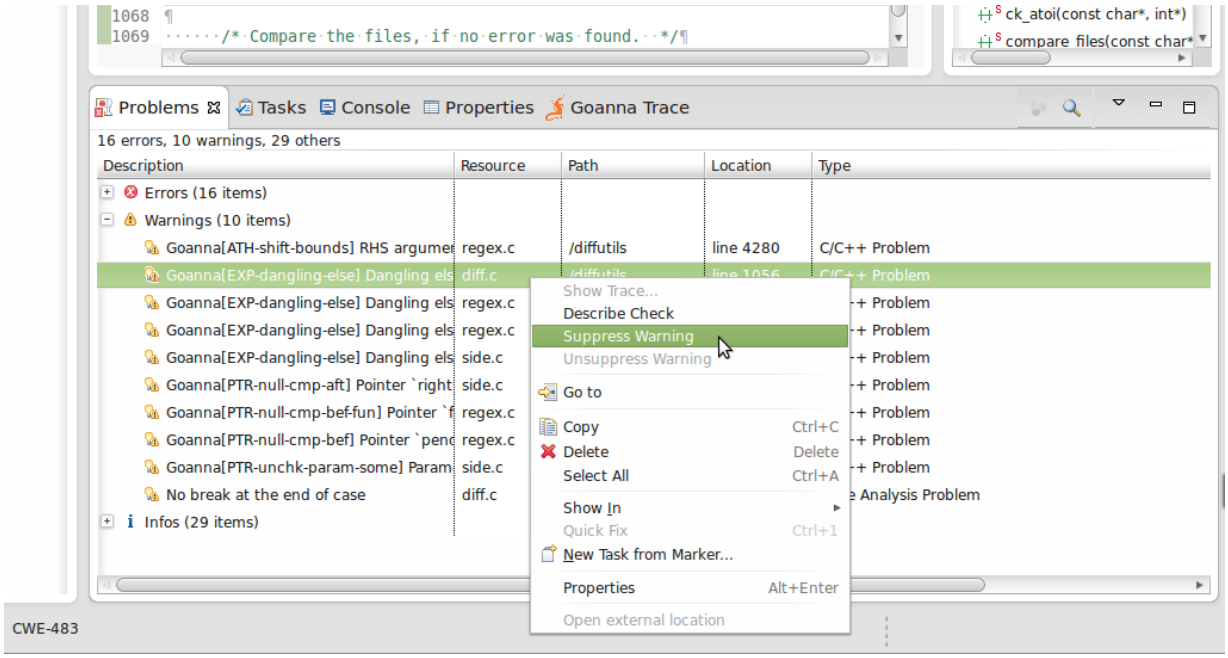
To enable the Warnings tab in Eclipse, select the Show Goanna Warnings in the Goanna drop-down menu. This will open a new tab Warnings which is basically what is displayed in Problems with some filtering. To act on the warnings, select the newly opened tab.

5.1 Suppressing Warnings

Quick Solution Right-click a warning and select Suppress Warning from the context menu. To view all suppressed warnings, click the Show Suppressed Warnings button on the right-hand side of the Goanna Warnings tab. From there, you can unsuppress warnings in the same way. □

To hide a warning, right-click it in the Goanna Warnings list, and select Suppress Warning from the context menu.

Goanna takes special care to ensure that warning suppression never silently conceals actual regressions (bugs that are fixed but inadvertently reintroduced) or unrelated bugs. If you suppress a warning, and later modify the function or any other code having an effect on it, Goanna will notice that the code has been edited, and re-issue the warning. Internally Goanna uses a hash of the actual syntax to determine if the code has changed, so only significant changes will cause the warning to reappear, not just modifying the file or changing whitespace or comments.



5.2 Showing Suppressed Warnings

When you ask Goanna Studio to ignore a warning, it will not be removed permanently from the database. The list of Goanna warnings in the Goanna Warnings tab can be toggled using the Toggle Suppressed Warnings, to switch the view from the reduced set of warnings (all but suppressed) to all warnings and back. Click the Show Suppressed Warnings button on the right-hand side of the Goanna Warnings tab to toggle which warnings are shown.

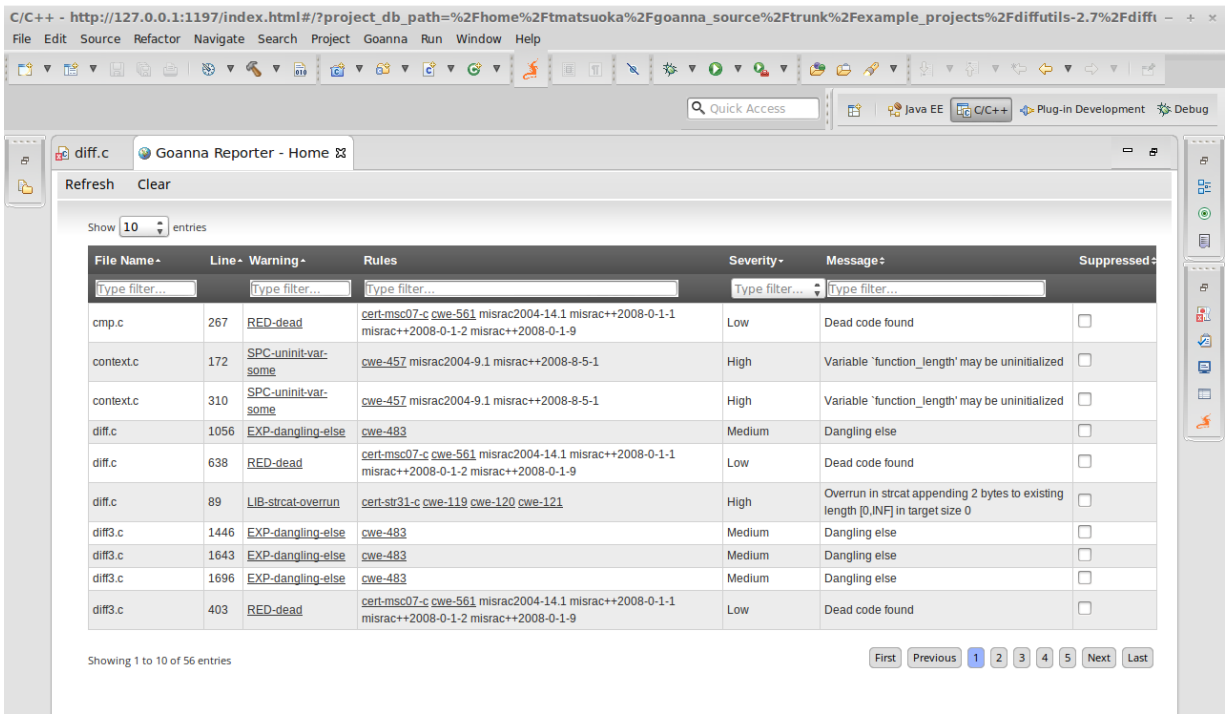
5.3 Unsuppressing Warnings

To unsuppress a warning, ensure you are showing suppressed warnings in the Warnings tab. Select the warning you wish to unsuppress, right-click on it, and select Unsuppress Warning from the context menu.

5.4 Warning Suppression Manager

The Warning Suppression Manager enables you to see the status of warnings and select which warnings to suppress or unsuppress using a simple web-based interface.

It allows you to search and sort the list of warnings for a selected project, as well as suppress and unsuppress warnings using the checkboxes on the right hand side of the screen.


C/C++ - http://127.0.0.1:1197/index.html#/?project_db_path=%2Fhome%2Ftmatsuoka%2Fgoanna_source%2Ftrunk%2Fexample_projects%2Fdiffutils-2.7%2Fdiff
File Edit Source Refactor Navigate Search Project Goanna Run Window Help
Quick Access
Java EE
C/C++
Plug-in Development
Debug
diff.c
Goanna Reporter - Home
Refresh
Clear
Show 10 entries
File Name
Line
Warning
Rules
Severity
Message
Suppressed
cmp.c 267 RED-dead cert-msc07-c cwe-561 misrac2004-14.1 misrac++2008-0-1-1 misrac++2008-0-1-2 misrac++2008-0-1-9 Low Dead code found
context.c 172 SPC-uninit-var-some cwe-457 misrac2004-9.1 misrac++2008-8-5-1 High Variable `function_length' may be uninitialized
context.c 310 SPC-uninit-var-some cwe-457 misrac2004-9.1 misrac++2008-8-5-1 High Variable `function_length' may be uninitialized
diff.c 1056 EXP-dangling-else cwe-483 Medium Dangling else
diff.c 638 RED-dead cert-msc07-c cwe-561 misrac2004-14.1 misrac++2008-0-1-1 misrac++2008-0-1-2 misrac++2008-0-1-9 Low Dead code found
diff.c 89 LIB-strcat-overrun cert-str31-c cwe-119 cwe-120 cwe-121 High Overrun in strcat appending 2 bytes to existing length [0,INF] in target size 0
diff3.c 1446 EXP-dangling-else cwe-483 Medium Dangling else
diff3.c 1643 EXP-dangling-else cwe-483 Medium Dangling else
diff3.c 1696 EXP-dangling-else cwe-483 Medium Dangling else
diff3.c 403 RED-dead cert-msc07-c cwe-561 misrac2004-14.1 misrac++2008-0-1-1 misrac++2008-0-1-2 misrac++2008-0-1-9 Low Dead code found
Showing 1 to 10 of 56 entries
First Previous 1 2 3 4 5 Next Last

5.4.1 Toggling Suppression Status

Under the Suppressed? column, you can check/uncheck the display status of a warning. Suppressed warnings will not be displayed in the future Goanna analysis.

5.4.2 Searching and Sorting Warnings

The Warning Suppression Manager provides powerful searching and sorting capabilities for warnings.

Clicking the column headers sorts the results alternately in ascending or descending order.

Each column also has a search box that is specific to the column. Search boxes accept any plain text keyword; for example, to search for all warnings from the check `SPC-return`, type `SPC-return` in the search box on the Warning column.

Clicking a check name in the Warning column will display the detailed description of that check. Additionally, some rules in the Rules column have links to the corresponding rule pages.

5.4.3 Other Features

Warnings can be exported in CSV format via the Export button.

If you make changes to the source files and then run analysis, new warnings will not automatically appear in the Warning Suppresion Manager. If this happens, click Refresh button to reload the warnings.

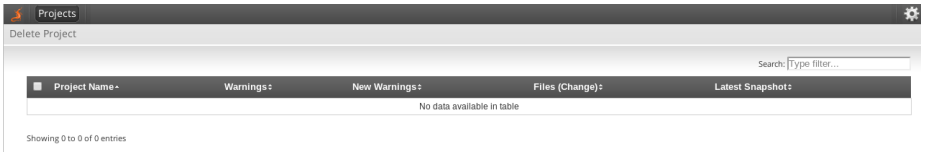
Clear button will delete all suppression, warnings and analysis information for that project. Solution and project options will remain.

6 Goanna Dashboard

Goanna Dashboard allows you to store and visualise the history of your Goanna results. It includes a web server, `goreporter`, used to display these results in a web browser. Each of your analysis runs is captured in a snapshot, which comprises of the warnings for that run as well as the source code analysed in the run. These snapshots are used to track the history of your project in the Goanna Dashboard.

6.1 Getting to the Goanna Dashboard

Click `Goanna Dashboard` in the Goanna menu to open the Goanna Dashboard. This will load the Project Page showing all projects that you have analysed with Goanna from Eclipse.



If you load the Goanna Dashboard without first taking a snapshot you will see an empty table with the text "No data available in table". To use the Goanna Dashboard you must first take a snapshot. (by running analysis, or using the "Take Snapshot" menu item)

6.2 Bug Statuses

The Goanna Dashboard allows you to classify bugs into one of five statuses:

Unclassified This is the default status for when a new warning is added to the Goanna Dashboard.

Ignore Ignore this warning; useful when the warning is valid, but does not require immediate attention.

Analyse This warning need to be investigated further before it can be classified properly.

Fix This warning is a problem that needs to be fixed.

Not a Problem This is warning is not a real bug (false positive).

If you select a warning to be either **Ignore** or **Not a Problem**, then these warnings are automatically *suppressed*; Goanna will then ignore these warnings in future analysis runs.

6.3 Severity

Every check in Goanna has been assigned a static severity of either; High, Medium or Low. These severities are represented throughout the Goanna Dashboard in three colours (or a blend of these three colours):

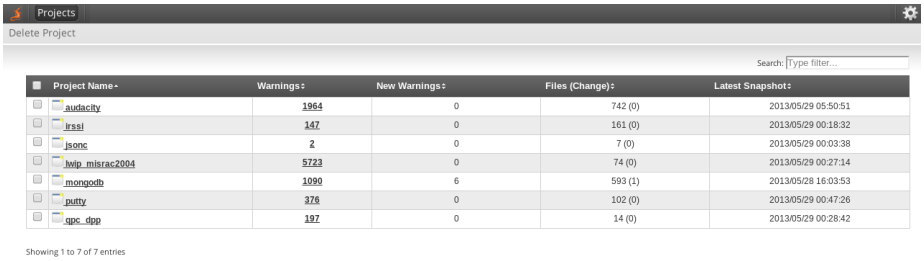
- **High:** Red
- **Medium:** Yellow
- **Low:** Green

6.4 Dashboard Views

6.4.1 Project Page

The project page is usually where you begin when using the Goanna Dashboard. It gives an overview of all projects Goanna has analysed and taken a snapshot of. The table provides a high level overview of the current state of all projects, including the number of warnings, number of new warnings, number of files analysed and the last time a snapshot was taken.

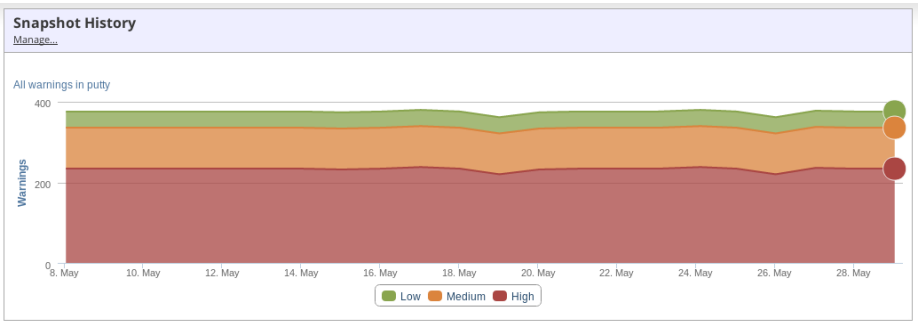
If you do not want to have a project in your dashboard anymore, you can use the checkboxes on the left hand side of the table to select the project(s) you wish to delete, then use the Delete Project button in the Dashboard's toolbar.



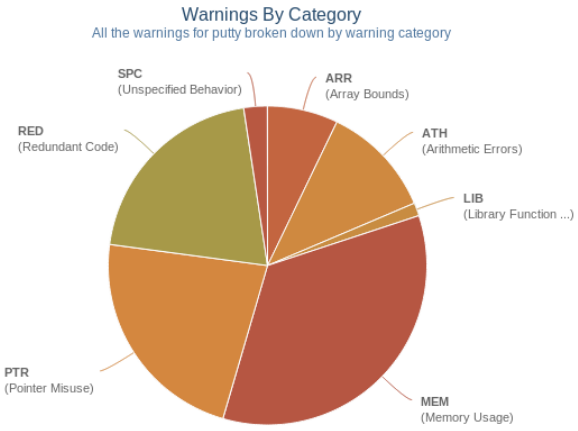
6.4.2 Report Page

Once you select a project you are taken to the project's report page. This page shows four graphs:

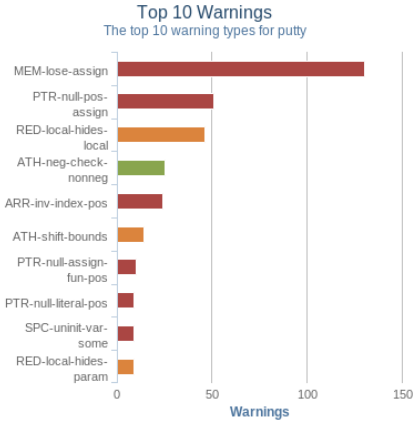
Snapshot History shows the overall progress of your project over time. Each point on the graph is one of your previous snapshots, showing the total number of warnings by severity. Clicking on any point in the graph will change the report page to show details about warnings in that snapshot. By default, the most recent snapshot is selected.



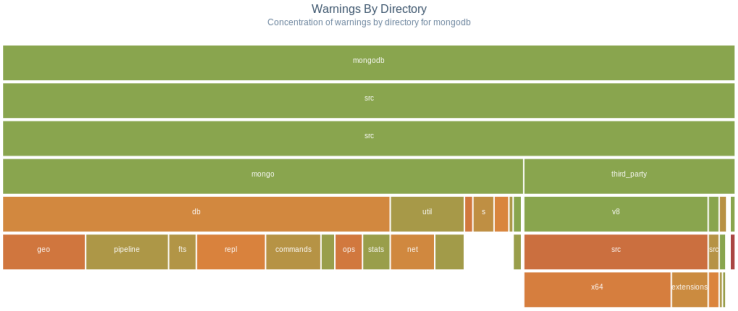
Warnings By Category shows all warnings for your snapshot broken down into each warning type. Clicking on a wedge will show the break down of warnings of that category. Clicking on a subsequent wedge will take you to the warnings browser filtered for that particular warning type.



Top 10 Warnings shows the top ten warning types in this snapshot. Clicking on one of the bars in the chart will take you to the warnings browser filtered by that warning type.

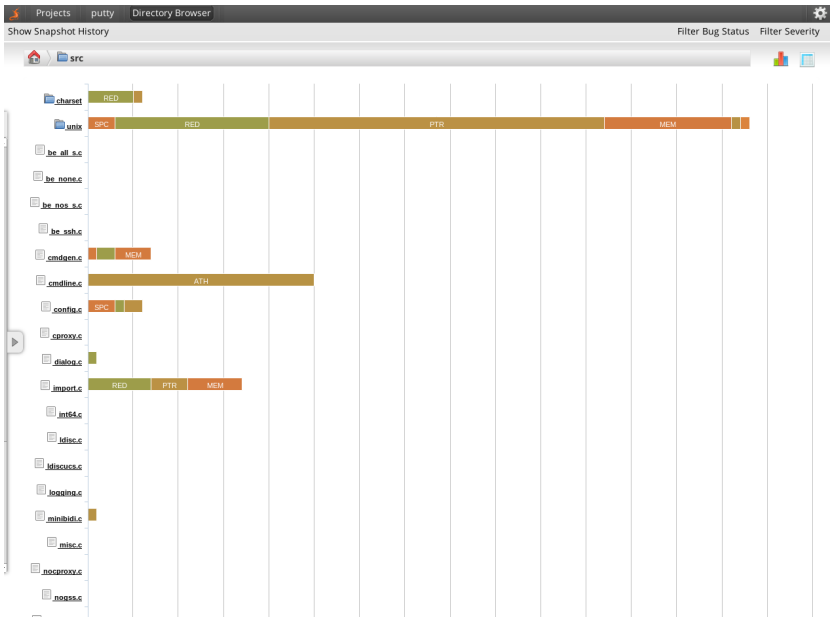


Warnings By Directory shows the concentration of warnings in your directory structure. A red node means that there are more than 10 warnings per file (average) in the directory, a green one means there are zero warnings per file (average). Clicking on a node will load the directory browser in that folder.



Global filters can be applied from the toolbar to filter by warning severity and/or bug status. These filters apply to all charts.

6.4.3 Directory Browser

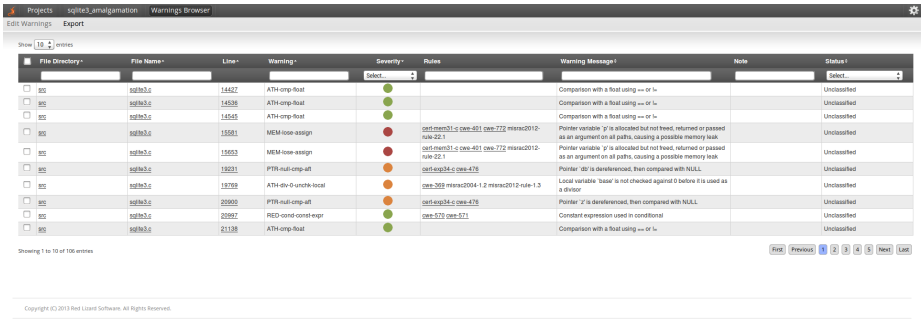


The directory browser is a way to browse through your project's directory structure to see what files and folders have what warnings. Total number of warnings per file or folder are broken down into warning categories and displayed in a bar chart.

The directory browser allows you to browse through your source tree in a few ways. The location bar above the chart allows you to see the path to your current location. Clicking on an item in the location bar will take you there in the directory browser. Similarly, clicking on a directory name in the chart will reload the directory browser with the contents of that folder. To view the contents of a source file, click on its name to load the code browser. To see the details about a particular warning category for a file, click on the segment for that category in the bar of the file or folder to load the warnings browser filtered for your selection.

The chart can be filtered in two ways. The sidebar allows for particular warning types, or warning categories to be turned on and off. In addition, the global filters in the toolbar (severity and bug status) also apply to this chart.

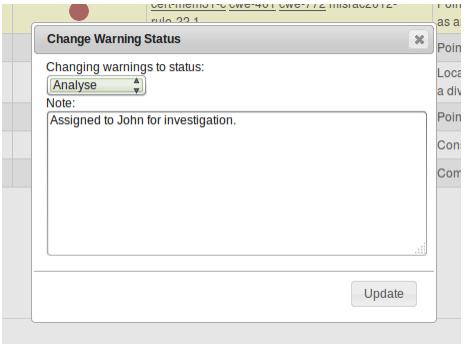
6.4.4 Warnings Browser



The warnings browser shows details of all the warnings in your project. Filtering is possible through the filter boxes in the header of the warnings table. The arrows in the table header allow for sorting.

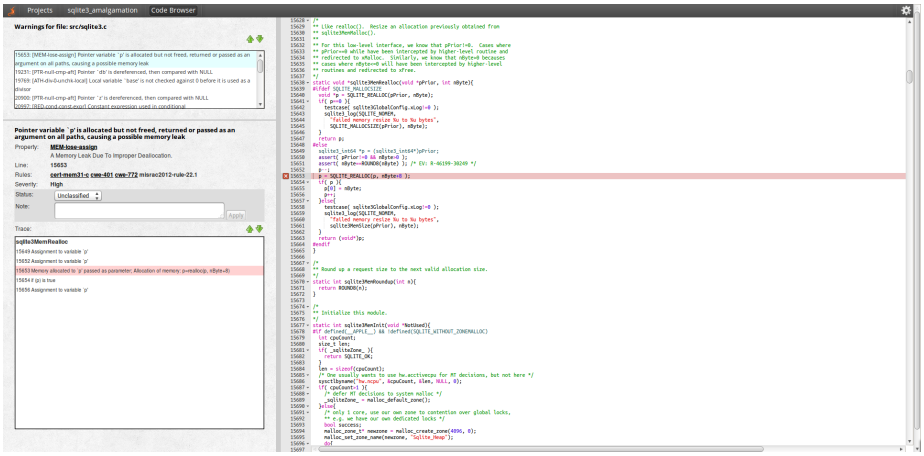
Clicking on a directory name will take you to the directory browser for that directory. A file name or line number will take you to the source code browser for that file and warning. Clicking on a Rule or Warning name will give you a description of that rule.

Selecting warnings then clicking "Edit Warnings" button opens a dialog where you can change their status and also add a note to the warnings. Clicking Update saves these changes.



The ‘Export’ button allows you to export all warnings or visible warnings to a CSV file.

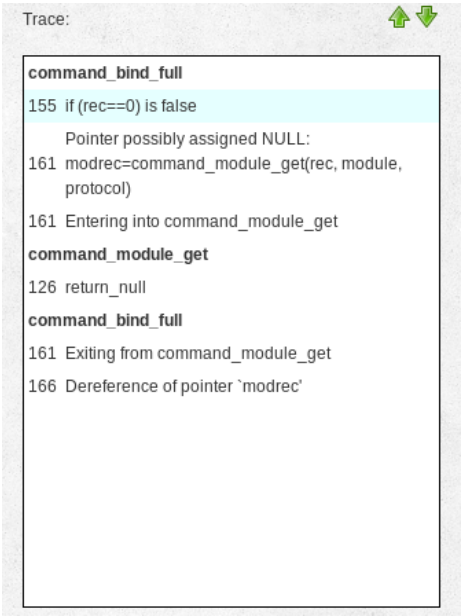
6.4.5 Code Browser



The code browser displays a file and all its Goanna warnings. The right hand side shows the source code of the file currently opened with warnings highlighted in three colours based on severity. The left hand side has the details of warnings for this file. The box at the top of the left hand pane allows you to select a warning to see in more detail.

Like in the warnings browser, you are able to change the status and add a note to each warning in the code browser. This can be done by changing the status and/or adding a note and pressing Apply.

If there is trace information for a warning it will also appear in this pane. You can step through the trace just like a debugger. To go to a step in the trace, click on it and the source code browser will jump to the corresponding line. You can then navigate through the trace using either the up/down arrows on top of the trace dialog, or by using the up/down arrow keys on your keyboard.

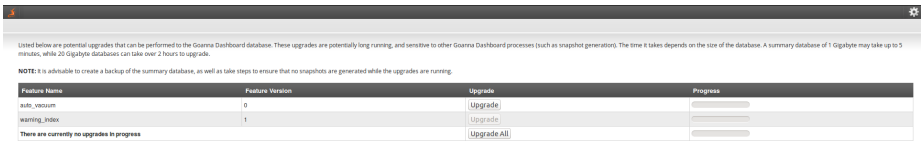


6.5 Database Upgrades

If you have upgraded from an older version of Goanna, Goanna may offer you to perform optional "database upgrades" to improve the performance of the Goanna Dashboard.

When you open Goanna Dashboard with the existing database, you may see the following notification:

Clicking the link will show a list of available optional upgrades:


Listed below are potential upgrades that can be performed to the Goanna Dashboard database. These upgrades are potentially long running, and sensitive to other Goanna Dashboard processes (such as snapshot generation). The time it takes depends on the size of the database. A summary database of 1 Gigabyte may take up to 5 minutes, while 20 Gigabyte databases can take over 2 hours to upgrade.
NOTE: It is advisable to create a backup of the summary database, as well as take steps to ensure that no snapshots are generated while the upgrades are running.
Feature Name
Feature Version
Upgrade
Progress
auto_vacuum
0
Upgrade
warning_index
1
Upgrade
There are currently no upgrades in progress
Upgrade All

In this version of Goanna, the following optional upgrades (called *features*) may be available:

- **`auto_vacuum`:** In Goanna 3.2.4 and before, deleting a snapshot or a project did not immediately delete corresponding data from the database. This may result in the database size to never shrink.

  Performing this upgrade causes Goanna to delete all residual data from removed snapshots and projects, and set up the database so that when a snapshot or a project is removed, Goanna removes corresponding data immediately. This ensures that the database size is always minimal.

  This upgrade was introduced in Goanna 3.3.0.
- **`warning_index`:** Performing this upgrade causes Goanna to apply optimizations to database indexes, resulting in improved performance, especially when loading Warnings Browser view.

  This upgrade was introduced in Goanna 3.3.0.

To apply an upgrade, click "Upgrade" button next to a desired feature. Alternatively, clicking "Upgrade All" button will apply all available upgrades.

Important Notes

We recommend that you take a backup of the database before applying any of the upgrade. The database is located in the following location:

`<root-of-workspace>/summary.goannadb`

You should not run Goanna analysis, interact with the Goanna Dashboard, or otherwise run any Goanna commands, until the upgrade is complete.

Performing upgrade will take a long time depending on the size of the database. If your database is significantly large, this may take hours to complete.

6.6 Project Settings (Advanced)

In general, the Goanna Dashboard should work for your project out of the box. However, there are a few advanced settings available if you want to customise the behaviour.

Settings are applied on a per-project basis. To access your project's settings click the gear menu in the top right hand corner after loading your project (through the project page), and select Project Settings.

6.6.1 Code Browser Character Encodings

By default the Dashboard will recognize source files in ASCII and UTF-8 (and additionally on Windows, the default character encoding used by your system). If your source files are not in any of these encodings, you will need to specify the encoding here in order for it to display correctly in the source code browser. A link is available in the Project Settings window listing all the encodings supported.

7 Command Line Options

This section is a detailed reference of the command line options that can be added to the *Extra arguments* field in your Project Settings.

7.1 Options

`--absolute-path` Print absolute paths in warnings.

`--all-checks` Run all available checks (overrides all other check related options).

`--brief-trace` Show immediately relevant decisions in trace output, not the majority of decisions.

`--c++` Indicate that file(s) contain C++ code.

`--check=<name>` Run a specific check (overrides any checks file).

`--checks-file=<file>` Use the checks listed in *<file>* instead of the default checks in `properties.init` file.

`--checks=<standard>` Run all checks in the specified coding standard. For example, `--checks=misrac2004` runs all available checks in the MISRA C:2004 standard.

`--checks=<standard>-<rule>` Run the check(s) corresponding to one rule in the specified coding standard. For example, `--checks=misrac2004-12.8` runs the check(s) that implement MISRA C:2004 rule 12.8.

`--color, --colour` Only available on Linux. Output in color.

`--columns` Print column positions in warnings.

`--db=<file>` Specify the database file to use for persistent information.

`--directory=<dir>` Before doing anything, change to *<dir>*.

`--force-analysis` Re-analyze files that have not changed since last run.

`--help` Print help message for common options.

`--ignore-errors` Ignore errors from the compiler.

`--license-server=<server[:port]>` Attempt to contact a license server at address *<server>*. *<Port>* is optional (defaults to 6200).

`--output-checks` Output the checks that are currently loaded.

`--output-format=<format>` Specify a warning format used by Goanna to output warnings. The following special strings in *<format>* are expanded:

- **`%FILENAME%`** the filename
- **`%RELFILE%`** the filepath and filename
- **`%RELPATH%`** the filepath
- **`%ABSFILE%`** the absolute filepath and filename
- **`%ABSPATH%`** the absolute filepath
- **`%DBRELFILE%`** the filepath relative to the database file and filename
- **`%DBRELPATH%`** the filepath relative to the database file
- **`%LINENO%`** the line number
- **`%COLUMN%`** the column number
- **`%CHECKNAME%`** the check identifier
- **`%SEVERITY%`** the checks severity rating
- **`%MESSAGE%`** the warning message
- **`%RULES%`** corresponding rule(s) from coding standards, if any
- **`%TRACE%`** counter example if any
- **`%FUNCTION%`** the function name
- **`%SUPPRESSED%`** a * if the warning is suppressed
- **`%WARNINGID%`** the hash of this warning
- **`%EOL%`** a line break
- **`%%`** a literal %.

The default warning format is: "`%RELFILE%:%LINENO%: warning: Goanna[%CHECKNAME%] Severity-%SEVERITY%, %MESSAGE%. %RULES%%EOL%`".

`--project-dir=<`*`path`*`>` Specify the location of the project root; default is `.`.

If the `--project-dir` option is given, then `--goanna-dir` is also required.

`--quiet` Only display warnings and no other output.

`--suppress=<`*`warning id`*`>` Suppress warning *<warning id>*.

`--suppression-status` Output suppression status markers, without suppressing warnings.

`--system-headers` Process system header files. Default: no.

`--timeout=<`*`n`*`>` Set a timeout (in seconds) for analysis of each source file. Default: 240.

Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

`--trace` Prints out a trace through the function that leads to the warning. This is helpful for understanding why the warning occurs.

`--trace-format=<`*`format`*`>` Specify the format to output traces. The following special strings are used in the trace format:

`%FILENAME%` the filename

`%RELFILE%` the filepath and filename

`%RELFILEX%` the filepath and filename followed by "`:`", or blank if in the current source file

`%RELPATH%` the filepath

`%ABSFILE%` the absolute filepath and filename

`%ABSFILEX%` the absolute filepath and filename followed by "`:`", or blank if in the current source file

`%ABSPATH%` the absolute filepath

`%DBRELFILE%` the filepath relative to the database file and filename

`%DBRELPATH%` the filepath relative to the database file

`%FUNCTION%` the function name

`%LINE%` the line number

`%TEXT%` text describing the event on the trace

`%TYPE%` the type of the trace line

`%EOL%` a line break

`%%` a literal %.

The default trace format is: "`%LINE%: %TYPE% - %TEXT%%EOL%`".

`--unsuppress=<`*`warning id`*`>` Unsuppress warning *<warning id>*.

`--user-headers` Process user header files. Default: no.

`--verbose` Display additional output information.

`--version` Print version information.

`--warning-ids` Output warning hashes.

`--with-cc=<`*`compiler`*`>` Specify the C compiler executable to run (if `--nc` is not specified). Also affects the default dialect when no `--dialect` is specified.

`--with-cxx=<`*`compiler`*`>` Specify the C++ compiler executable to run (if `--nc` is not specified). Also affects the default dialect when no `--dialect` is specified.

Advanced Command Line Options

The following options are intended to be used only in cases where your environment requires them. In general, you do not need to use these options.

`--32` Analyze code for 32-bit targets (longs and pointers are 32 bits wide).

`--64` Analyze code for 64-bit targets (longs and pointers are 64 bits wide).

`--advanced-help` Print help message for advanced options.

`--dialect=<file>` Specify the dialect of C/C++ compilers. Available dialects are:

```
armcc       metrowerks
c166        microsoft
c51         qnx
cc21k       renesas-h8
cosmic      renesas-rx
diab        ti-armcl
gnu         ti-cl16x
iar-8051    ti-cl2000
iar-arm     ti-cl430
iar-avr     ti-cl470
iar-msp430  ti-cl500
iar-m16c    ti-cl55
```

If you use this option, you should also specify `--with-cc`, `--with-cxx` and/or `--with-ld` to specify the paths to the compiler(s) and linker. If these are not specified, then Goanna will assume the default name for the specified dialect, which may not be what is available on your system.

If none of `--dialect`, `--with-cc`, `--with-cxx` or `--with-ld` are specified, then Goanna will assume the default of `gnu` dialect with `gcc` C compiler, `g++` C++ compiler and `ld` linker.

`--error` Exit with error status code when warnings emitted.

`--exclude=<file>` Exclude the specified *<file>* from analysis.

`--html-report=<output type>` After analysis, also generate analysis report files in HTML format. You can optionally specify type of HTML reports to be generated ('summary', 'warnings' or 'all'). `--html-report` with no type will generate all available reports.

`--input-encoding=<type>` Specify the character encoding of the source file:

us-ascii ASCII (default)

utf-8 UTF-8

ansi (Available on Windows only) default character encoding of the system

`--internal-error=<value>` Exit with *<value>* on internal error.

`--ipa-iterations=<value>` Specify the number of times interprocedural analysis iterates towards a fixed point. The default is 2.

Important: Setting this value to 0 (meaning keep iterating until a fixed point is reached) is discouraged; this may cause Goanna to not terminate!

`--ipa-trace-depth=<value>` How many levels of inlining are performed for interprocedural traces. Default: 5.

Important: Setting this value to -1 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

`--issue-report=<type>` Control generation of issue report files:

never Never

on-failure On failures only

on-error On failures and analysis errors

timeout On failures, errors, and timeouts

always Always (even if successful)

`--license-borrow-hours=<number>` When contacting license server, borrow license for *<number>* of hours. Defaults to 1, maximum of 24.

`--license-dir=<directory>` Set directory in which to look for a license file.

`--no-globals` Do not analyze global integer variables.

`--no-ipa` Disable interprocedural analysis.

`--output-file=<file>` Append warning messages to a specified file.

`--output-spec=<file>` Use the contents of *<file>* as the output-format.

`--output-xml=<file>` After analysis, also output warnings in XML format to *<file>*.

`--parse-error-log=<file>` Log parse errors to the specified file instead of stderr.

`--timeout-error=<value>` Exit with status code *<value>* when too many timeouts occur.

`--timeout-limit=<value>` Maximum number of per-phase timeouts. Default: 3.

Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

`--timeout-per-phase=<n>` Set a timeout (in seconds) for each phase of analysis. This is useful if you have a few functions that take very long to analyze and you would like to limit the time spent on these, while still getting as many results as possible on everything else. Default: 60.

Important: Setting this value to 0 (meaning infinite) is discouraged; this may cause Goanna to not terminate!

Diagnostics Command Line Options

The following options are provided for diagnostics purposes only. **Do not use these options unless directed by Red Lizard Software support team.**

`--alias`

`--configure=<dialect>`

`--dialect-mod=<dialect-mod>`

`--dataflow`

`--diagnostics-mode`

`--no-alias`

`--no-dataflow`

`--no-default-packages`

`--package=<package>`

`--package-dir=<directory>`

`--record`

Any unrecognized options are treated as compiler arguments.

8 Support

This section describes a number of known issues and solutions to them. Should you find more issues you think we should know about, please contact us at mailto:support@redlizards.com.

8.1 Known Issues

The following section describes some known issues present in this version of Goanna Studio. In addition to these, there are also some known issues that apply to some compilers - see 1.4 for details.

8.1.1 Known Issues For All Goanna Studio Eclipse Edition Users

Project References

Goanna may not recognize some build configurations if your project references another project.

Workspace on Network Drives

Windows only: Goanna Dashboard and Suppression Manager may fail to start if your workspace is located in a network drive.

8.1.2 Known Issues For QNX Momentics

Goanna may fail to run if your project is a QNX C Project or QNX C++ Project.

8.1.3 Known Issues For Texas Instruments Code Composer Studio

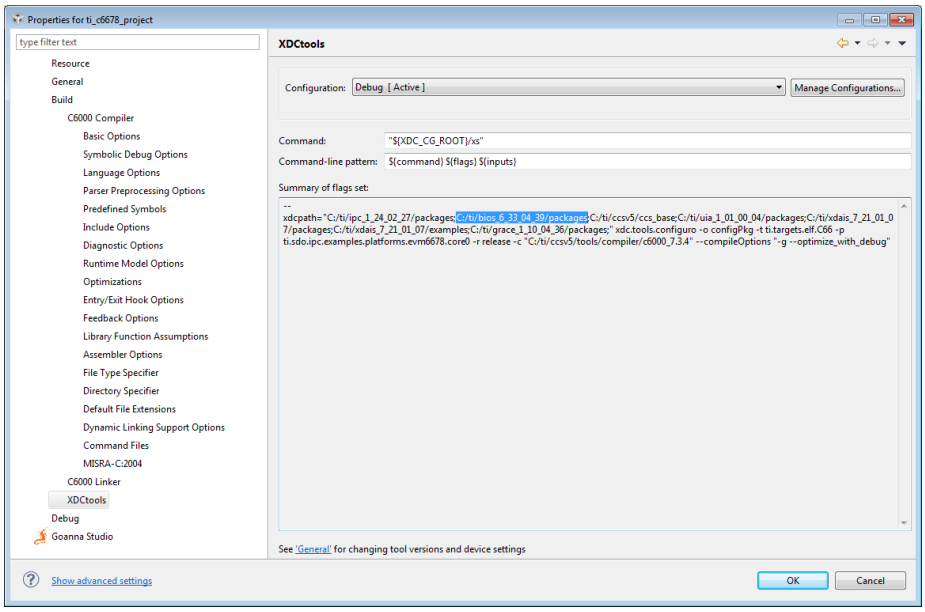
eXpress DSP Components (XDC) tools header include paths are not automatically detected. If your project uses these header files, Goanna may fail with parse errors.

If this is the case, then you will need to add XDC tools header include paths into "Extra Arguments" text box in Goanna Project Properties. The include paths comprise of:

- XDC package path, and
- The remaining include paths.

By default the XDC package path is in the format of `C:\ti\xdctools_<version>\packages`, where *<version>* is the version of XDCtools your project is using in the format similar to, for example, `3_25_03_72` (for version 3.25.3.72). To find the version of XDCtools you are using, click Project Properties, then open General page, and then RTSC tab.

To find the remaining XDC include paths you are using, click Project Properties and open the XDCtools properties page. The include paths are in "Summary of flags set" list, and in the format of **`-xdcpath="<include paths>"`**.



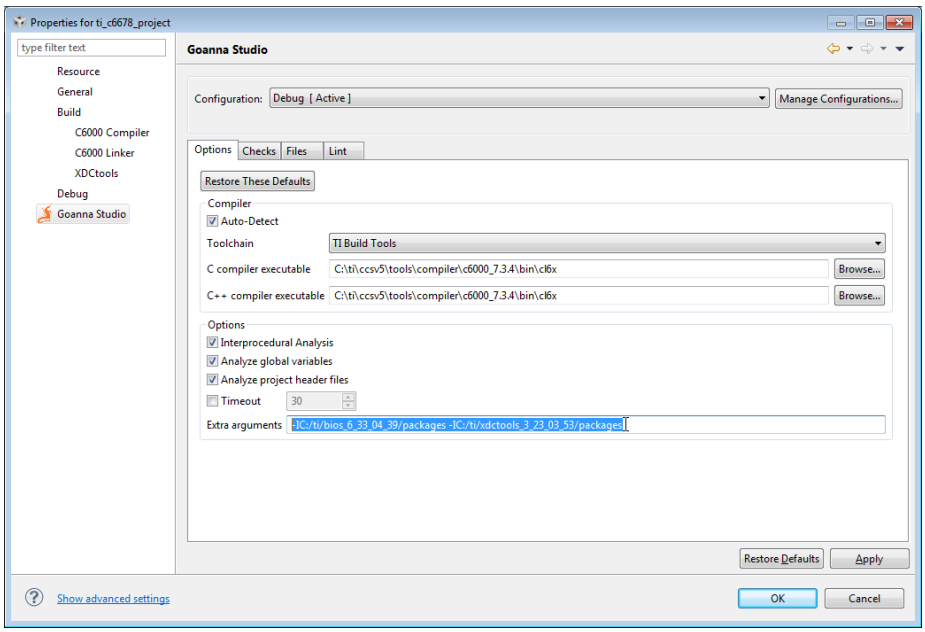
The include path options you need to pass to goanna would be in the format of:

`-I<XDC package path> -I<first path from XDCtools dialog> -I<second path from XDCtools dialog> ...`

For example,

`-IC:/ti/xdctools_3_25_03_72/packages -IC:/ti/bios_6_33_04_39/packages -IC:/ti/ccsv5/ccs_base`

Input the include path options composed in the format above into "Extra arguments" entry of Goanna Project Properties.



8.1.4 Known Issues For Wind River Workbench

Installation Paths

Goanna only supports Wind River Workbench if Wind River Workbench installed to the default installation path of `C:\WindRiver`. For VxWorks project, Goanna only supports VxWorks 6.9 (installed to `C:\WindRiver\vxworks-6.9`).

Temporary Header Files

Goanna may not be able to find a VxWorks header file `ipcom_config.h`, because by default this file is a temporary header file that is only available during build, not analysis.

If this is the case, you will need to open:

`C:\WindRiver\components\ip_net2-6.9\osconfig\vxworks\make\rules.ipcom`

and comment out the following line:

```
.INTERMEDIATE : \$(_IPCOM_IPDOMAIN_FILE_)
```

and rebuild your project before running Goanna analysis.

Index

`_GOANNA` preprocessor symbol, 30
`assert` macro, 30

Command Line Options
- --32, 41
- --64, 41
- --absolute-path, 40
- --advanced-help, 41
- --alias, 43
- --all-checks, 40
- --brief-trace, 40
- --c++, 40
- --check, 40
- --checks, 40
- --checks-file, 40
- --color, --colour, 40
- --columns, 40
- --configure, 43
- --dataflow, 43
- --db, 40
- --diagnostics-mode, 43
- --dialect, 41, 42
- --dialect-mod, 43
- --directory, 40
- --error, 42
- --exclude, 42
- --force-analysis, 40
- --goanna-dir, 41
- --help, 40
- --html-report, 42
- --ignore-errors, 40
- --input-encoding, 42
- --internal-error, 42
- --ipa-iterations, 29, 42
- --ipa-trace-depth, 42
- --issue-report, 42
- --license-borrow-hours, 42
- --license-dir, 42
- --license-server, 40
- --nc, 41
- --no-alias, 43
- --no-dataflow, 43
- --no-default-packages, 43
- --no-globals, 42
- --no-ipa, 29, 42
- --output-checks, 40
- --output-file, 42
- --output-format, 40
- --output-spec, 42
- --output-xml, 42
- --package, 43
- --package-dir, 43
- --parse-error-log, 42
- --project-dir, 41
- --quiet, 41
- --record, 43
- --suppress, 41
- --suppression-status, 41
- --system-headers, 41
- --timeout, 41
- --timeout-error, 43
- --timeout-limit, 43
- --timeout-per-phase, 43
- --trace, 41
- --trace-format, 41
- --unsuppress, 41
- --user-headers, 41
- --verbose, 41
- --version, 41
- --warning-ids, 41
- --with-cc, 41
- --with-cxx, 41

Common Weakness Enumeration (CWE), 22
Computer Emergency Response Team (CERT) C/C++ Coding Standard, 22

Dialect
- Available dialects, 20

False positives, 24, 29
- Bug status, 33

Goanna Dashboard, 33
- Bug status, 33
- Views, 33

Interprocedural analysis, 29
Issue report, 26

License, 14
- activation, 14
- floating, 15

Motor Industry Software Reliability Association (MISRA) C++:2008, 22
Motor Industry Software Reliability Association (MISRA) C:2004, 22

Standards, 22

Traces
- Dashboard, 37

Warning
- Suppression, 33, 36